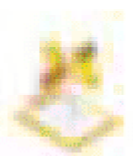
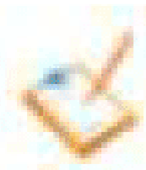
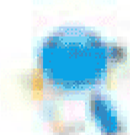


 新手入门  逐步进阶  实战提高

 图解教学  范例练习  视频光盘



HTML 5与CSS 3
网页设计入门与提高

张惠芳 徐小慧 编著

影响百万人的经典清华版
全新改版震撼上市





光盘超值赠送
全部案例的源文件
多媒体视频演示

清华大学出版社

软件入门与提高丛书

HTML 5 与 CSS 3 网页设计入门与提高

张惠芳　徐小慧　编著

清华大学出版社
北 京

内 容 简 介

本书从初学者的角度出发，由浅入深、循序渐进地介绍了 HTML 5 和 CSS 3 应用与开发的相关知识，书中提供了大量操作 HTML 5 和 CSS 3 新增功能的示例，还提供了用于演练的实战和上机练习。

本书共分为 15 章，主要内容包括 HTML 文档结构、文档基础标记、列表标记、表格标记和表单标记，CSS 发展历史、基本语法和常用样式，JavaScript 脚本的基本语法、变量、运算符、语句类型、对象和函数，网页设计流程和网页布局，HTML 5 的发展历史、使用 HTML 5 的原因、HTML 5 的语法，新增的结构元素、分组元素、文本语义元素、交互元素、音频和视频元素、标准属性，新增表单元素、新增输入类型、新增表单属性、表单验证，canvas 元素及其 API 绘图，文件操作和文件拖放、客户端存储数据、本地数据库、跨文档传输信息、多线程、获取位置信息，CSS 3 新增颜色、新增选择器、新增文件属性、字体属性、背景属性、边框属性、盒模型、页面布局，以及渐变、转换、过渡和动画等内容。在本书最后一章利用 HTML 5 和 CSS 3 等技术实现一个案例作为结束。

本书几乎涉及 HTML 5 和 CSS 3 应用与开发的所有重要知识，适合所有的 HTML 5 和 CSS 3 初学者进行学习。另外，对于大中专和培训班的学生来说，本书更是一本不可多得的教材。

图书在版编目(CIP)数据

HTML 5 与 CSS 3 网页设计入门与提高/张惠芳，徐小慧编著. --北京：清华大学出版社，2015
(软件入门与提高丛书)
ISBN 978-7-302-38455-7
Ⅰ. ①H…　Ⅱ. ①张…　②徐…　Ⅲ. ①超文本标记语言—程序设计　②网页制作工具　Ⅳ. ①TP312
②TP393.092

中国版本图书馆 CIP 数据核字(2014)第 260930 号

责任编辑：张　瑜　杨作梅
装帧设计：刘孝琼
责任校对：马素伟
责任印制：
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：28.5　　**字　数**：689 千字
(附 DVD 1 张)
版　　次：2015 年 1 月第 1 版　　**印　次**：2015 年 1 月第 1 次印刷
印　　数：1～3000
定　　价：59.00 元

产品编号：057494-01

前　　言

将 HTML、CSS 和 JavaScript 结合使用是一种最常用的网页布局，HTML 即超文本标记语言，它使用标记来描述网页。HTML 5 是 HTML 早期版本的自然延续，它尽可能地满足了当前网站和未来网站的需求。HTML 5 从以前的版本中继承了大部分特性，这就意味着，HTML 5 的大部分内容都可以兼容新旧浏览器，向后兼容是 HTML 5 的一项重要设计原则。

CSS 在 1996 年正式推出，HTML 诞生几年之后才出现了 CSS 的第一个版本，CSS 是指层叠样式表，它的样式定义了如何显示 HTML 网页中的元素。CSS 3 是 CSS 早期版本的自然延续，它比早期版本更为强大，引入了大量的视觉效果，例如阴影、文字阴影、圆角和渐变等。

本书详细介绍 HTML 5 和 CSS 3 的新增知识，在介绍这些新增功能的同时，还会介绍 HTML 和 CSS 早期版本的一些内容，以及 JavaScript 的基础知识。

1. 本书内容

全书共分 15 章，主要内容如下。

第 1 章：HTML 快速入门。首先向读者介绍 HTML 的概念及其发展历史，然后重点讲解 HTML 4 的文档结构及其提供的标记。

第 2 章：CSS 基础。首先向读者介绍 CSS 的概念及其发展历史，然后重点介绍 CSS 2 的常用语法和常用样式。

第 3 章：JavaScript 脚本语言。从 JavaScript 脚本的概念开始介绍，然后依次介绍脚本的基础语法、变量、运算符、语句类型和常用对象等内容。

第 4 章：网页设计实战案例。向读者介绍实际网页设计时需要掌握的各种技能。包括网页设计流程、网页设计工具、网页布局以及布局理论等内容。

第 5 章：认识 HTML 5。着重介绍 HTML 5 的知识，包括它的发展历史和趋势、三大组织、基本语法、新增的表单、元素以及属性等多种内容。

第 6 章：HTML 5 快速入门。介绍 HTML 5 中新增的不同类型的元素，例如结构元素、分组元素、文本语义元素、交互元素、音频和视频元素等。另外，还对 HTML 5 中常用的几个标准属性进行介绍。

第 7 章：HTML 5 新型表单的使用。从表单开始介绍，接着介绍 HTML 5 中新增的表单元素、输入类型、表单属性和表单验证这 4 个知识点。

第 8 章：HTML 5 操作页面图形。重点介绍 canvas 元素及其 API 如何绘制图形并对图形进行操作。这些图形包括文本、矩形、线条、圆形和扇形、贝塞尔曲线、线性渐变、径向渐变以及图像等。

第 9 章：HTML 5 的其他新特性。从文件新增特性、拖放功能、新增客户端数据存储特性、新增的本地数据库特性、跨文档传输信息、多线程以及获取位置信息 7 个方面介绍

HTML 5 的新增特性。

第 10 章：CSS 3 快速入门。介绍 CSS 3 的基础知识，包括 CSS 3 的发展、优缺点和浏览器支持情况，以及 CSS 3 的颜色、选择器和属性等新增功能。其中对新增颜色进行了详细介绍。

第 11 章：CSS 3 新增选择器。从属性选择器、结构化伪类选择器、目标伪类选择器、UI 元素状态伪类选择器、否定伪类选择器和通用兄弟选择器 6 个方面进行介绍。

第 12 章：CSS 3 页面美化样式。重点介绍 CSS 3 中新增的与文本、字体、背景和边框有关的样式属性。

第 13 章：CSS 3 页面布局样式。从多列布局、盒模型和界面布局三个方面详细介绍 CSS 3 新增的属性、语法格式及其使用示例。

第 14 章：CSS 3 动画特效。首先了解 CSS 3 中如何实现渐变，接着介绍 CSS 3 中新增的转换功能，然后介绍与过渡有关的属性，最后对 CSS 3 的动画功能进行说明。

第 15 章：HTML 5 + CSS 3 页面案例。结合 HTML 5、CSS 5、JavaScript 和 jQuery 等多种技术实现贪吃蛇游戏。

2. 本书特色

本书内容详细、示例丰富、知识面广，全面地讲解了 HTML 5 和 CSS 3 的应用和开发。与已经出版的同类图书相比，这本图书的最大特点体现在如下几个方面。

(1) 知识全面，内容丰富

本书紧密围绕 HTML 5 和 CSS 3 的新增知识展开详细的讲解，涵盖了实际开发应用中的具体应用代码。

(2) 理论和示例结合

本书中几乎每一个知识点都有丰富而典型的练习，而且每一章最后都会通过一个或多个综合的实战介绍本章的知识。作为一本 HTML 5 和 CSS 3 入门类型的书，作者把理论和练习很好地结合起来进行讲解，最容易让读者快速掌握。

(3) 应用广泛，提供文档

对于大多数的精选实战案例，都会向读者提供详细的实现步骤，结构清晰简明，分析深入浅出，而且有些实战贴近实际。

(4) 随书光盘

本书配备了视频教学文件，包括每个章节所涉及的源代码、开发环境的安装演示等。读者可以通过视频文件更加直观地学习 HTML 5 和 CSS 3 的知识。

(5) 网站技术支持

读者在学习或者工作的过程中，如果遇到实际问题，可以直接登录 www.itzcn.com 与我们取得联系，作者会在第一时间内给予帮助。

(6) 贴心的提示

为了便于读者阅读，全书还穿插着一些技巧、提示等小贴士，体例约定如下。

- 提示：通常是一些贴心的提醒，让读者加深印象或提供建议和解决问题的方法。
- 注意：提出学习过程中需要特别注意的一些知识点和内容，或者相关信息。
- 技巧：通过简短的文字，指出知识点在应用时的一些小窍门。

3. 读者对象

本书适合作为软件开发入门者的自学用书，也适合作为高等院校相关专业的教学参考书，还可供开发人员查阅、参考。

主要读者对象包括：

- HTML 5 和 CSS 3 开发入门者。
- HTML 5 和 CSS 初学者以及在校学生。
- 各大、中专院校的在校学生和相关授课老师。
- 准备从事与 HTML 5 和 CSS 3 应用相关的工作人员。

除了封面署名作者之外，参与本书编写的人员还有程朝斌、王咏梅、郝军启、王慧、郑小营、张浩华、王超英、张凡、赵振方、张艳梅等，在此表示感谢。

在本书的编写过程中，我们力求精益求精，但难免存在一些不足之处，敬请广大读者批评指正。

编　者

目　　录



第 1 章

HTML快速入门

HTML 是 Hypertext Markup Language(超文本标记语言)的简称，它通过标记符号来标记要显示的网页中的各个部分。HTML 语言是制作网页的基础语言。作为一个网页制作爱好者或者专业的网页设计师，HTML 语言知识是不可或缺的。因此，在学习 HTML 5 技术之前，掌握 HTML 的基础是非常必要的。

本章首先向读者介绍 HTML 的概念及其发展历史，然后重点讲解 HTML 4 的文档结构及其提供的标记。

本章学习目标：

- 了解 HTML 的发展历史
- 掌握 HTML 文档结构中每个标记的使用
- 掌握元信息、字体、超链接、水平线标记和段落标记的使用
- 掌握编号列表、符号列表和说明列表标记的使用
- 掌握定义表格的方法
- 掌握表格、行和列的常用属性
- 掌握创建表单的方法
- 熟悉常用的表单元素

1.1 HTML 的概念

HTML 用标记来表示网页中的文本及图像等元素，并规定浏览器如何显示这些元素，及如何响应用户的行为，是标准通用标记语言(Standard Generalized Markup Language，SGML)的一种应用。

HTML 是在 1989 年被 Web 的发明者提出的，它的标准由万维网协会(W3C)负责制定，还推出了 DHTML(动态 HTML)和 XML 语言。所有的网页都是以 HTML 格式的文件作为基础，再加上一些其他的语言构成的。

(1) HTML 2.0

HTML 2.0 是 1996 年由 Internet 工程工作小组的 HTML 工作组开发的。

(2) HTML 3.2

HTML 3.2 作为 W3C 标准，发布于 1997 年 1 月 14 日。HTML 3.2 向 HTML 2.0 标准添加了被广泛运用的特性，如字体、表格、Applet、围绕图像的文本流、上标和下标等。

(3) HTML 4.0

作为 W3C 推荐的一项标准，HTML 4.0 发布于 1997 年 12 月 18 日。而仅仅进行了一些编辑修正的第二个版本发布于 1998 年 4 月 24 日。HTML 4.0 最重要的特性是引入了样式表(CSS)。

(4) HTML 4.01

作为 W3C 推荐的一项标准，HTML 4.01 发布于 1999 年 12 月 24 日。HTML 4.01 是对 HTML 4.0 的一次较小的更新，对后者进行了修正和漏洞修复。W3C 不想继续发展 HTML，打算把未来的工作集中在 XHTML 上。

(5) XHTML 1.0(可扩展的超文本标记语言)

XHTML 1.0 使用 XML 对 HTML 4.01 进行了重新表示。

作为 W3C 推荐的一项标准，XHTML 1.0 发布于 2000 年 1 月 20 日。XHTML 1.0 是作为一种 XML 应用被重新定义的 HTML，其目标是取代 HTML 4.01。

XHTML 1.0 与 HTML 4.01 最大的区别就是语法要求更加严格，所有标记由 DTD 规则来定义，而且可以自定义标记，文档的结构也更清晰。

(6) HTML 5

为了推动 Web 标准化的发展，一些互联网公司联合起来成立了 WHATWG 组织。WHATWG 组织致力于 Web 表单和应用程序，而 W3C 专注于 XHTML 2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。

HTML 5 草案的前身名为 Web Applications 1.0，它于 2004 年被 WHATWG 提出，于 2007 年被 W3C 接纳，并成立了新的 HTML 工作团队。

HTML 5 的第一份正式草案已于 2008 年 1 月 22 日公布，最新草案公布于 2010 年 6 月 24 日。虽然 HTML 5 仍然处于完善之中，但是大部分主流的浏览器已经具备了 HTML 5 的运行环境。因此，可以说 HTML 5 将引领 Web 发展到一个新的高度，并掀起一轮学习 HTML 5 的热潮。

1.2 HTML 的文档结构

开发人员使用 HTML 所编写的超文本文档称为 HTML 文档，它能独立于各种操作系统平台。

HTML 有自己的语法格式和编写规范，这些都是由 HTML 规范所定义的，下面详细介绍 HTML 文档的规范及结构。

1.2.1 文档编写规范

HTML 文档是由一系列的元素和标记组成的。一个完整的 HTML 文档至少应该包括<!DOCTYPE>标记、html 元素、head 元素、title 元素和 body 元素。元素名不区分大小写；标记用于规定元素的属性和它在文档中的位置。

1. HTML标记

HTML 标记可分为成对标记和单独标记两种。多数标记成对出现，由开始标记和结束标记组成。开始标记的格式为“<元素名称>”，结束标记的格式为“</元素名称>”，其完整语法格式如下：

```
<元素名称>元素内容</元素名称>
```

成对标记仅对包含在其中的元素内容发生作用，如<title>和</title>标记用于定义页面标题元素的范围。也就是说，<title>和</title>标记之间的部分是此 HTML 文档的标题。

单独标记的格式为“<元素名称>”，其作用是在相应的位置插入元素。例如<hr>标记便是在该标记所在位置插入一个水平线。

在 HTML 标记中还可以设置一些属性，控制 HTML 标记所建立的元素。这些属性将位于所建立元素的开始标记中，其基本语法格式如下：

```
<元素名称 属性 1="值 1" 属性 2="值 2" ...>
```

因此，在 HTML 文档中某个元素的完整定义语法如下：

```
<元素名称 属性 1="值 1" 属性 2="值 2" ...>元素内容</元素名称>
```

2. HTML元素

当用一组 HTML 标记将一段文字包含在中间时，这段包含文字的 HTML 标记被称为一个元素。在 HTML 语法中，每个由 HTML 标记与文档所形成的元素内，还可以包含另一个元素。因此整个 HTML 文档就像是一个大元素包含了许多小元素。

在所有的 HTML 文档中，最外层的元素由<html>标记建立。在<html>标记所建立的元素中包含了两个主要的标记，这两个标记是<head>标记和<body>标记。<head>标记所建立的元素内容为头部元素，而<body>所建立的元素内容为主体元素。

如下所示为一个 HTML 文档的标准结构：

```
<!DOCTYPE>
<html>
    <head>头部元素</head>
    <body>
        主体元素
    </body>
</html>
```

1.2.2 文档声明标记

<!DOCTYPE>标记用于定义有关文档格式的声明，它必须写在 HTML 文件中的第一行。使用格式如下：

```
<!DOCTYPE element-name DTD-type DTD-name DTD-url>
```

上述格式说明如下。

- <!DOCTYPE：表示开始声明 DTD，其中 DOCTYPE 是关键字。
- element-name：指定该 DTD 的根元素名称。
- DTD-type：指定该 DTD 是属于标准公用的还是私人制定的，若设为 PUBLIC 则表示该 DTD 是标准公用的，如果设为 SYSTEM 则表示是私人制定的。
- DTD-name：指定该 DTD 的文件名称。
- DTD-url：指定该 DTD 文件所在的 URL 网址。
- >：表示 DTD 的结束声明。

下面列出了如何引用 W3C 在 HTML 4.01 版本和 XHTML 中所制定的几种 DTD。具体如下所示。

(1) HTML 4.01 Strict DTD：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```

(2) HTML 4.01 Transitional DTD：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
```

(3) HTML 4.01 Frameset DTD：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
```

(4) XHTML 1.0 Strict：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

(5) XHTML 1.0 Transitional：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

(6) XHTML 1.0 Frameset：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

(7) XHTML 1.1：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

(8) XHTML 1.1 plus MathML plus SVG：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG
1.1//EN" "http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg.dtd">
```

在上述声明设置中，Strict(严格型)版本只允许使用 XHTML，除非 Web 开发者打算只依赖样式表而不使用任何格式的标记来编写 XHTML 文档，否则不推荐使用这个规范。Transitional(过渡型)版本是最灵活的规范，它允许用户使用不被推荐但仍然可用的元素和属性。Frameset(框架型)版本与 Transitional 版本类似，但是它支持需要使用框架的元素。

在 HTML 文件中，所有的控制标记必须以 html 为根标记，所以在 DTD 的声明中 element-name 必须是“HTML”或“html”。因为在 HTML 规范中，对于标记中的英文大小写是不区分的，但在 XHTML 文件中 element-name 必须是“html”。

当软件(如浏览器)解析 HTML 文件时，如果需要这些 DTD，则可以通过 DTD-url 指定的网址去下载这些 DTD。这 3 个 DTD 都是经过 W3C 认证的，所以属于公认的标准。由于 HTML 文件不允许引用其他自定义的 DTD，所以 DTD-type 的设置必须是 PUBLIC。

提示：为什么我们在绝大多数的 HTML 文件中都没有设置 DTD 声明呢？这是因为浏览器在解读 HTML 文件时，都是使用浏览器本身指定的 DTD，所以就不再去下载标准的 DTD 了。这是浏览器可以使用自己扩展控制标记的原因，也是 HTML 文件没有设置 DTD 的声明仍可以被浏览的原因。但如果 HTML 文件要被其他软件解读的话，一定要加上 DTD 的声明，否则该软件就无法判断该 HTML 文件是否是有效的文件。

1.2.3 标记文档开始

文档标记以<html>标记开始，以</html>标记结束，用于表示该文档是以超文本标记语言(HTML)编写的。

其语法格式如下：

```
<html>文档的全部内容</html>
```

任何一个规范的 HTML 文档最先出现的 HTML 标记都是<html>，而且它必须成对出现。开始标记<html>和结束标记</html>分别位于 HTML 文档的最开始和结束，文档中的所有 HTML 标记都包含在<html>标记中。

事实上，常用的 Web 浏览器(如 IE)都可以自动识别 HTML 文档，并不要求有<html>标记，也不对该标记进行任何操作。但是，为了提高文档的适用性，使编写的 HTML 文档能适应不断变化的 Web 浏览器，应当养成使用这个标记的习惯。

1.2.4 标记文档头部

HTML 文档头部以<head>标记开始，以</head>标记结束。HTML 文档头部用于定义文档的标题(出现在 Web 浏览器窗口的标题栏中)和一些属性。标题包含在<title>和</title>标记之间，其语法格式如下：

```
<head>
    <title>文档标题</title>
</head>
```

在<head>标记所定义的元素中不允许放置网页的任何内容，而是放置关于 HTML 文档的信息。也就是说，它并不属于 HTML 文档的主体，它包含文档的标题、编码方式及 URL 等信息，这些信息大部分是用来提供索引和其他方面的应用的。

每个 HTML 文档都需要有一个文档名称。在浏览器中，文档名称作为窗口名称显示在该窗口的最上方，这对浏览器的收藏功能很有帮助。HTML 文档的标题要写在<title>与</title>标记之间，该组标记应包含在<head>与</head>标记当中。

提示：HTML 文档的标记是可以嵌套的，即在一对标记中可以嵌入另一对子标记。嵌套在<head>标记中的常用标记有<title>、<meta>和<style>。

1.2.5 标记文档主体

HTML 文档主体部分以<body>标记开始，以</body>标记结束。HTML 文档主体用于存放页面中要显示的文本、图像和链接。其语法格式如下：

```
<body>主体部分</body>
```

<body>标记是成对出现的，网页中的主体内容应该写在<body>与</body>之间，而<body>标记包含在<html>与</html>标记之间。

1.2.6 编写注意事项

在编写 HTML 文档时，要注意以下事项。

(1) 尖括号“<”和“>”是任何标记的开始和结束，元素的标记要用这对尖括号括起来，并且结束标记在形式上总是在开始标记前加一个斜杠(/)。

(2) 标记之间可以嵌套，例如：

```
<body>
    <center>
        <div>初始 HTML 文档</div>
    </center>
</body>
```

(3) HTML 元素是不区分大小写的。例如，以下几种写法都是正确的，并且表示相同的标记：

```
<HEAD>
<Head>
<head>
```

(4) 任何回车和空格在 HTML 代码中都不起作用。为了代码清晰，建议不同的标记之间在回车换行后编写。

(5) HTML 标记中可以放置各种属性，如下面的代码所示：

```
<div align="center">唐诗 300 首</div>
```

其中，align 为<div>标记的属性，center 为 align 属性的值。

注意：属性应该出现在“<>”内，并且与元素名之间应该用一个空格分开，属性值应该使用引号括住。

(6) 如果需要在 HTML 文档代码中添加注释，可以用“<!--”开始，以“-->”结束。例如下面的代码：

```
<!-- 文档范例 1-2 -->
<!-- 文档说明：第一个 HTML 文档 -->
<html>
    ...
</html>
```

注释语句只出现在 HTML 文档代码中，而不会在浏览器中显示出来。

1.2.7 实战——创建第一个HTML文档

在详细了解 HTML 文档的每个组成部分之后，本小节将创建一个非常简单的 HTML 文档，演示 HTML 文档的创建方法。具体步骤如下。

(1) 创建一个名为 welcom.html 的文件，用记事本打开，进行编辑。

(2) 在第一行使用!DOCTYPE 指令添加 XHTML 1.0 的声明代码，如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

(3) 创建一个标准的 HTML 文档结构，即<html>标记中包含<head>和<body>两个子标记。代码如下：

```
<html>
<head></head>
<body></body>
</html>
```

(4) 向<head>标记内添加一个<meta>标记，设置文档的编码为 utf-8。代码如下：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

(5) 使用<title>标记设置文档的标题为“我的第一个 HTML 页面”。代码如下：

```
<title>我的第一个 HTML 页面</title>
```

(6) 向<body>标记中添加页面需要显示的主体内容。代码如下：

```
<h1>春暖花开</h1>
<hr/>
<p>春暖花开，这是我的世界</p>
<p>每次怒放，都是心中喷发的爱</p>
<hr/>
<p align="right">更多内容请看这里<a href="#">春暖花开</a></p>
```

上述代码中使用的各种 HTML 标记将在后面介绍。

(7) 向文档中添加几个注释。如下所示为本示例的最终代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>我的第一个 HTML 页面</title>
</head>
<!--以上为 HTML 的头部-->
<body>
<!--这里是 HTML 的主体-->
<h1>春暖花开</h1>
<hr/>
<p>春暖花开，这是我的世界</p>
<p>每次怒放，都是心中喷发的爱</p>
<hr/>
<p align="right">更多内容请看这里<a href="#">春暖花开</a></p>
</body>
</html>
<!--HTML 文档结束-->
```

(8) 用 Chrome 浏览器打开 welcome.html 文件，查看示例运行的效果，如图 1-1 所示。在图 1-1 中，并没有显示注释，可以右击，从弹出的快捷菜单中选择“查看源代码”命令查看 HTML 源代码，如图 1-2 所示。



图 1-1 示例运行的效果



图 1-2 查看源代码

1.3 文档基础标记

从文档结构中可以看出，HTML 页面是由很多标记组成的。本节将介绍 HTML 中基础标记的使用，如使用元信息标记定义页面关键字、使用字体标记定义字体大小和颜色等。

1.3.1 元信息标记

元信息使用的是<meta>标记，它必须放在 HTML 文档的<head>和</head>标记内。<meta>标记不需要结束标记，在标记内的是一个元信息内容。

<meta>标记常用的属性有两个：name 和 http-equiv。其中 name 用于描述网页，以便于搜索引擎查找和分类。常用的语法如下：

```
<meta name="keyname" content="具体的关键字" />
<meta http-equiv="name" content="内容" />
```

用户可以使用 meta 来设置不同的内容，如页面关键字、页面描述、作者信息，以及页面的定时跳转等。

【例 1.1】

创建一个示例，分别使用<meta>标记的 name 和 http-equiv 属性指定页面的元信息。

(1) 使用 name 和 content 设置页面关键字。代码如下所示：

```
<meta name="keyword" content="教程,视频教程,编程视频" />
```

keyword 表示要设置的是页面关键字，具体关键字由 content 属性来指定，多个关键字之间使用逗号分隔。上面的代码设置了 3 个关键字，分别是教程、视频教程和编程视频。

(2) 设置页面描述也是为了便于搜索引擎的查找。与关键字一样，设置的页面描述不会在网页中显示出来。代码如下：

```
<meta name="description" content="窗内网用户注册页面" />
```

(3) 通过指定 name 和 content 的值，可以在页面的源代码中显示页面制作者的姓名以及个人信息。代码如下：

```
<meta name="author" content="itzcn.com" />
```

(4) 如下代码限制了搜索引擎对页面的搜索方式：

```
<meta name="robots" content="index|all|nofollow|noindex|none" />
```

上述代码给出了 content 属性所有可能的值，其中 index 表示只能搜索当前页，all 表示能搜索当前网页及其链接的网页，nofollow 表示不能搜索当前网页链接的网页，noindex 表示不能搜索当前网页，none 则表示不能搜索当前网页及其链接的网页。

(5) 使用 meta 还可以实现自动刷新页面的功能。如下代码中，页面 5 秒刷新一次：

```
<meta http-equiv="refresh" content="5" />
```

上述代码中 refresh 表示网页的刷新，content 属性的内容指定页面刷新的时间，默认

情况下，刷新时间以秒为单位。

(6) 将 http-equiv 属性的值指定为 expires，表示设置网页的到期时间，一旦过期，必须到服务器上重新调用。到期时间必须是 GMT 时间格式，即“星期，日 月 年 时 分 秒”，它们都使用英文和数字指定。代码如下：

```
<meta http-equiv="expires" content="Wed,14 september 2013 15:30:30 GMT" />
```

(7) 将 http-equiv 属性的值指定为 charset，表示设置网页的文档编码。如下代码设置编码为 utf-8：

```
<meta http-equiv="charset" content="utf-8" />
```

除了 utf-8 编码之外，常用的编码还有 gbk、gb2312 和 iso-8859-1。

1.3.2 字体标记

一个完整的 HTML 网页少不了文本内容(即字体)，HTML 把用来显示和处理的标记称为字体标记，它可以对文字样式、颜色和字体大小进行设计。HTML 中与字体有关的标记有<hn>、<font>、<strong>、<em>和<small>等。

1. <hn>标记

<hn>标记通常也会被称为标题文字标记，每一种标题在字号上都有明显的区别。n 的值可以是整数 1~6 中的任何一个，从 1 级到 6 级逐渐减小。该标记的使用非常简单，以 1 级标题为例，语法如下：

```
<h1>...</h1>
```

【例 1.2】

在新建的项目中添加全新的 HTML 页面，在该页面中依次添加 h1 到 h6 的标记。代码如下：

```
<h1>推荐：HTML 5 系列视频教程</h1>
<h2>网页设计的最佳工具</h2>
<h3>HTML 网页设计经典案例</h3>
<h4>快速建站实用指南</h4>
<h5>如何使用搜索引擎</h5>
<h6>原创模板下载</h6>
```

运行页面查看效果，如图 1-3 所示。

默认情况下，<hn>标记的标题文字是靠左对齐的。而在实际的网页设计页面中，需要将标题文字放置在不同的位置，最常用的就是使用 align 属性。align 属性的值有 3 个：left、center 和 right。

【例 1.3】

对图 1-3 中的标题指定 align 属性来更改显示位置。代码如下：

```
<h1 align="center">推荐：HTML 5 系列视频教程</h1>
<h2 align="left">网页设计的最佳工具</h2>
<h3 align="right"> HTML 网页设计经典案例</h3>
```



图 1-3　<hn>标记练习

重新运行页面，具体效果不再给出，请读者思考。

2. <font>标记

<font>标记是 HTML 网页中最常用的一个标记，它用来表示文字样式，通过向该标记中添加属性，可以实现各种各样的文字效果。<font>标记的语法如下：

```
<font face="字体样式" size="字体大小" color="字体颜色">文字内容</font>
```

在上述语法形式中，可以直接使用<font>而不添加任何属性，也可以添加一个或多个属性。常用属性的说明如下。

(1) face 属性

face 属性可以设置文字的不同字体效果，例如，将字体设置为“黑体”、“隶书”、“宋体”以及“华文彩云”等。face 属性的值可以有一个或多个，多个属性值之间使用逗号分隔。默认情况下使用第 1 种字体进行显示。如果第 1 种字体不存在，则使用第 2 种字体进行代替，以此类推。如下代码展示了 face 属性：

```
<font face="黑体,宋体,新宋体">我的世界</font>
```

注意：用户设置字体的效果时，必须确保用户系统中已经安装了相应的字体，否则这些字体会被浏览器中的普通字体所代替。因此，用户在设计网页时应当尽量不使用特殊字体，以免浏览页面时无法看到正确的效果。

(2) size 属性

size 属性是指字体的大小，它没有一个相对的大小标准，其大小只是相对于默认字体而言。size 属性的有效范围值是整数 1~7，其中 3 是默认值。用户在使用时可以在 size 属性前添加“+”或“-”符号，以指定相对于默认字体大小的增量或减量。

【例 1.4】

如下代码展示了 size 属性设置不同字体大小的用法：

```
<font face="黑体,宋体,新宋体" size="5">春天</font>
<font face="黑体,宋体,新宋体" size="+7">夏天</font>
```

```
<font face="黑体,宋体,新宋体" size="-4">秋天</font>
<font face="黑体,宋体,新宋体" size="2">冬天</font>
```

(3) color 属性

color 属性用于设置字体的颜色，该属性的值可以是关键字 red、yellow、black、green 和 blue 等。但是，大多数情况下 color 属性是通过十六进制的颜色代码(RGB)表示的。如表 1-1 列出了常用字体颜色的名称和十六进制颜色值。

表 1-1　常用字体颜色名称和十六进制颜色值

颜色名称	十六进制颜色值	颜色名称	十六进制颜色值
black(黑色)	#000000	olive(橄榄色)	#808000
white(白色)	#FFFFFF	red(红色)	#FF0000
yellow(黄色)	#FFFF00	maroon(栗色)	#800000
aqua(青色)	#00FFFF	teal(深青色)	#008080
blue(蓝色)	#0000FF	navy(深蓝色)	#000080
gray(灰色)	#808080	silver(浅灰色)	#C0C0C0
green(绿色)	#00800	lime(浅绿色)	#00FF00
purple(紫色)	#800080	fuchsia(紫红色)	#FF00FF

【例 1.5】

如下代码展示了 color 属性在<font>标记中的简单使用：

```
<font color="red">红色</font>
<font color="blue">蓝色</font>
<font color="#FFFFFF">白色</font>
<font color="#00FFFF">青色</font>
```

通常情况下，不会将某个标记的属性单独使用，而是将这些标记和属性结合起来使用，下面通过一个简单的练习进行介绍。

【例 1.6】

重新更改例 1.2 中的代码，为网页中的文字设置效果。代码如下：

```
<font size="3" color="red"  face="黑体">HTML 网页设计经典案例</font>
<br/>
<font size="+3" color="#000000" face="黑体, Courier New">推荐：HTML 5 系列
视频教程</font>
<br/>
<font size="+1" color="#000080" face="新宋体">网页设计的最佳工具</font>
<br/>
<font size="-2" color="#FF00FF" face="宋体">快速建站实用指南</font>
<br/>
<font size="5" color="#808080" face="Times New Roman">如何使用搜索引擎
</font>
<br/>
<font size="4" color="#00FFFF" face="楷体">原创模板下载</font>
```

重新运行页面来查看效果，如图 1-4 所示。



图 1-4　字体标记的使用

3. 其他标记

<font>标记的 face、size 和 color 属性虽然可以完成对字体的绚丽设计，但是，有些情况下，这些设计并不能满足用户的需求，这时可以借助其他的一些标记。这些标记能够让文字有更多的样式变化，也可以强调某一部分，如表 1-2 所示对常用的 HTML 其他字体标记进行了说明。

表 1-2　HTML 中的其他常用字体标记

字体标记	含　义
	粗体
<i></i>	斜体
<u></u>	下划线
<tt></tt>	打字机字体
<big></big>	大型字体
	小型字体
	表示强调，一般为斜体
	表示特别强调，一般为粗体

【例 1.7】

如下代码展示了这些常用标记的简单使用，开发人员向页面添加代码，完成后可以运行页面查看效果。代码如下：

```
<b>常见的体育运动</b>
<i>作者：张艾</i>
<u>通过 u 标记添加下划线</u>
<big>big 标记显示大型字体</big>
<small>small 标记表示小型字体</small>
<em>em 标记显示 Welcome</em>
<strong>粗体</strong>
```

1.3.3 超链接标记

超链接(Hyperlink)是超级链接的简称，它是网页中最重要的元素之一。可以按照使用对象的不同，将网页中的链接进行分类，如文本超链接、图像超链接、E-mail 链接、锚点链接、多媒体文件链接和空链接等。

链接是从一个 Web 页到另一个相关 Web 页的有效途径，在 HTML 文档中通过<a>标记来实现超链接。当浏览网页时，单击一个超链接，可使网页切换到另一个 HTML 文档或 URL 指定的站点。<a>标记的一般格式如下：

```
<a href="链接地址">超链接说明文字</a>
```

上述语法格式中，href 属性所指向的链接地址可以是本地计算机上的一个文件，也可以是某个站点或网页中的 URL，还可以是本网页中的一个书签；“超链接说明文字”是网页中链接上显示的文字。

<a>标记中的链接地址可以采用绝对路径和相对路径两种引用，绝对路径是主页上的文件或目录硬盘上的真正路径(例如 F:\MyWork\NewHtml.htm)；相对路径适合于网站的内部链接(例如 InnerHtml_2.htm)。下面给出相对链接常用的 3 种方式：

- 如果链接到同一个目录下，只需输入要链接的文档的名称，如 name.htm 或 name.html。
- 如果链接到下一级目录中的文件，只需先输入目录名，然后再加“/”和文件名，如/work/name.htm 或/work/name.html。
- 如果要链接到上一级目录中的文件，则先输入“../”，然后输入目录名和文件名，例如../work/name.htm 或../work/name.html。

【例 1.8】

创建一个示例，为<a>标记中的 href 属性指定不同的值，并单击网页中的链接信息进行测试，步骤如下。

步骤01 在网页的合适位置添加<p>标记，在该标记中嵌套<i>标记和<a>标记，将<a>标记的 href 的值指定为窗内网的网址 www.itzcn.com，代码如下：

```
<p>链接到窗内网站点：<i><a href="http://www.itzcn.com">进入窗内网</a></i></p>
```

步骤02 继续添加<a>标记，为 href 属性的值使用相对路径链接到当前项目根目录下的 index.html 页面。代码如下：

```
<p>链接到首页：<i><a href="/index.html">站点首页</a></i></p>
```

步骤03 添加<a>标记，链接到一个电子邮箱。代码如下：

```
<p>链接到电子邮箱：<i><a href="mailto:798804212@qq.com;">写信给我</a></i></p>
```

步骤04 添加<a>标记，设置 href 属性的值，链接到本地路径。如下所示：

```
<p>链接到本地的帮助文档页面：<i><a href="E:\MyName\help.html">查看帮助</a></i></p>
```

步骤05 运行页面查看效果，如图 1-5 所示。可以单击测试链接功能。



图 1-5　页面运行效果

默认情况下，用户单击页面中的超链接时，会覆盖当前的页面。有时候，用户并不希望新打开的网页窗口(即目标窗口)将原来的窗口覆盖，这时可以通过设置 target 属性来设置目标窗口的显示位置。

扩展<a>标记的语法如下所示：

```
<a href="链接地址" target="_parent | _blank | _self | _top">超链接说明文字</a>
```

从上述语法内容中可以看到，target 属性的常用值有 4 个：_parent、_blank、_self 和 _top。表 1-3 对这 4 个值进行了说明。

表 1-3　target 属性的常用属性值

target 属性值	说　明
_parent	在上一级窗口中打开，常在分帧框架页面中使用
_blank	浏览器总在一个新打开、未命名的窗口中载入目标文档
_self	默认设置，在同一个窗口中打开目标文档
_top	在浏览器的整个窗口中打开，将会忽略所有的框架结构

提示： target 属性的 4 个值都是以下划线开始，任何其他用一个下划线作为开头的窗口或者目标都会被浏览器所忽略。因此，不要将下划线作为文档中定义的任何框架 name 或 id 的第一个字条。

【例 1.9】

我们重新更改例 1.8 中的代码，为每个超链接标记添加 target 属性，并指定 target 属性的值。

以第一个标记为例，代码如下：

```
<p>链接到窗内网站点：<i><a href="http://www.itzcn.com" target="_blank">进入
窗内网</a></i></p>
```

重新运行页面，单击链接内容进行测试，打开的窗口网页效果如图 1-6 所示。



图 1-6　打开新的窗口网页

1.3.4　水平线标记

水平线用于段落和段落之间的分隔，它使文档结构更加清晰明了，文字的编排更加整齐。HTML 中，水平线标记是<hr>，它是独立使用的标记符(即没有结束标记符)，用来绘制一条自动换行的水平直线，直线的上下两端都会有一定的空白。

<hr>标记也有许多自身属性，常用属性说明如下。

- width：该属性用于设置水平线的长度，它的取值可以是具体的数值(单位是像素)，也可以是百分比数值。当使用后者时，表示占浏览器窗口的百分比。
- height：用于设置水平线的高度，它的取值可以是具体的数值(单位是像素)，也可以是百分比数值。
- size：该属性用于设置水平线的粗细，单位为像素。
- align：用于设置水平线的对齐方式。其属性值可以使用 left(左对齐)、right(右对齐)和 center(居中，默认值)中的任意一个。
- color：设置水平线的颜色。

【例 1.10】

创建一个 HTML 页面，使用<hr>标记为页面添加 3 条水平线。步骤如下。

步骤01　第一条直接使用<hr/>标记，不使用任何其他属性。代码如下：

```
<hr/>
```

步骤02　在页面中使用<hr>标记绘制一个宽度是 220、粗细是 3、颜色值是 FFCC00 的水平线，并且将该水平线居中。代码如下：

```
<hr width="220px" align="center" size="3" color="#FFCC00" />
```

步骤03　使用<hr>标记绘制一个粗细是 2、颜色值是 33FFFF 的水平线。代码如下：

```
<hr size="2" color="#33FFFF" />
```

步骤04　运行 HTML 页面，查看上面绘制的三条水平线的效果，如图 1-7 所示。



图 1-7　使用水平线标记

1.3.5 段落标记

网页中的文字结合起来，就形成了一个段落，HTML 中与段落有关的标记有多个，如换行标记、段落标记、居中对齐标记和缩进标记等。下面详细介绍每个段落标记的具体使用方法。

1. 换行标记

表示一个回车符，浏览器遇到该标记时会产生一个换行，该标记没有对应的结束标记。使用
标记实现的文本效果能够使用文本以比较紧凑的形式显示。一个
标记代表一个换行，连续的多个标记可以多次换行。

和
都可以达到换行的效果。简单的使用示例见下面的代码：

```
一年有 4 个季节。<br><br>
一年有 12 个月。<br />一年有 365 天，每天 24 小时。
```

2. 段落标记

<p>表示段落标记，在<p></p>之间的内容属于同一个文本段落。我们可以使用成对的<p>标记来包含段落，也可以单独使用<p>标记来划分段落。在<p>标记中可以使用 align 属性说明该段落文本的对齐方式，它的值可以是 left(左对齐)、right(右对齐)或者 center(居中对齐)。

【例 1.11】

用<p>标记将页面分为两个段落，在段落中使用
标记进行换行处理。代码如下：

```
<p>    爱上美色，只是因为爱上那种纯净里的万种风情。<br />
    每个不同性格、爱好的人都会找到自己的最爱。而打动他们的可能
只是每个鸡尾酒背后的美丽传说和魅力特征。有人列了一份清单：</p>
<p>    相爱的恋人，一份“激情海滩”与“巧克力马天尼”酒能使浓
情蜜意流露无遗。<br /> <br />
    单身的男人，一份“黑色伏特加”尽现成熟稳健的气息。<br />
<br />
    年轻女孩在人群中穿梭，捧一杯“蓝色夏威夷”，杯中闪动的耀目
```

```
剔透的海蓝色，在黑夜渲染着天蓝的清朗，在所有人心中留下美丽的痕迹。<br /> <br />
    粉红色的“玛格丽特”散放冰冻草莓香，有着调酒师纪念女友的动
人传说。</p>
```

上述代码中， 表示添加一个空白空格，所有内容添加完成后，运行网页，效果如图 1-8 所示。



图 1-8　使用段落标记

3. 居中对齐标记

通过<p>标记的 align 属性，可以将段落中的内容居中。此外，HTML 中还提供了专门的对齐标记<center>。该标记的使用非常简单，代码如下：

```
<center>文字内容</center>
```

【例 1.12】

修改例 1.11 中的代码，在<p>标记的两端添加<center>标记，如下所示：

```
<center>
    <!--省略内容-->
</center>
```

<center>标记会自动将文字居中显示，运行效果如图 1-9 所示。



图 1-9　段落内容居中

4. 缩进标记

使用<blockquote>标记可以实现页面文字的段落缩进。每使用该标记一次，段落就会被缩进一次，可以嵌套使用该标记，以达到不同的缩进效果。

<blockquote>标记的语法如下：

```
<blockquote>段落文字</blockquote>
```

【例 1.13】

在 HTML 页面中添加 3 个<blockquote>标记，并在该标记中使用<p>标记定义缩进段落的内容，代码如下：

```
<blockquote>
   <blockquote>
     <p>爱上美色，只是因为爱上那种纯净里的万种风情。<br />
     <blockquote>每个不同性格、爱好的人都会找到自己的最爱。而打动他们的可能只是每个鸡尾酒背后的美丽传说和魅力特征。有人列了一份清单： </blockquote>
     </p>
     <p>相爱的恋人，一份“激情海滩”与“巧克力马天尼”酒能使浓情蜜意流露无遗。</p>
   </blockquote>
   <p>单身的男人，一份“黑色伏特加”尽现成熟稳健的气息。<br />     <br />
     年轻女孩在人群中穿梭，捧一杯“蓝色夏威夷”，杯中闪动的耀目剔透的海蓝色，在黑夜渲染着天蓝的清朗，在所有人心中留下美丽的痕迹。<br /> <br />
      粉红色的“玛格丽特”散放冰冻草莓香，有着调酒师纪念女友的动人传说。 </p>
</blockquote>
```

运行上述代码查看效果，如图 1-10 所示。



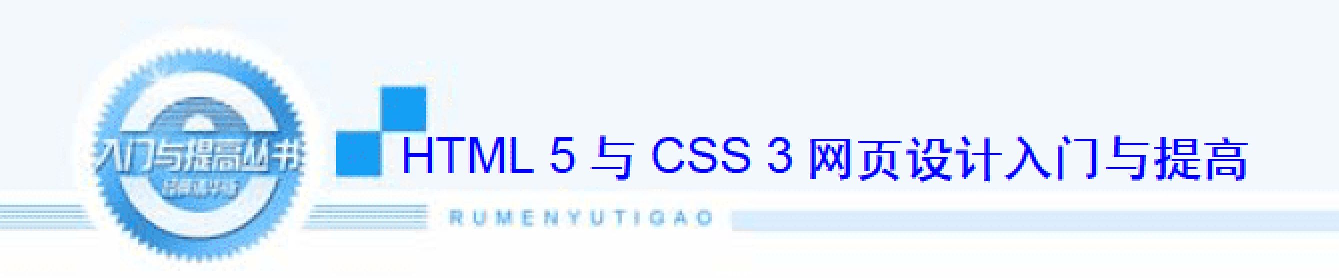
图 1-10 使用段落缩进标记

1.4 列表标记

HTML 支持 3 种类型的列表标记，分别是编号列表、项目符号列表和说明项目列表标记。下面通过不同的示例来介绍这些列表的用法。

1.4.1 编号列表

编号列表使用<ol>标记，每一个列表项前使用<li>，每个项目都有前后顺序之分。编



号列表标记的语法如下：

```
<ol>
    <li>列表项 1</li>
    <li>列表项 2</li>
    <li>列表项 3</li>
    ...
</ol>
```

标记还具有 type 属性和 start 属性。其中，type 属性用于设置编号的种类；start 为编号的开始序号。type 属性的取值及含义如下。

- 1：表示序号为数字。
- A：表示序号为大写英文字母。
- a：表示序号为小写英文字母。
- I：表示序号为大写罗马数字。
- i：表示序号为小写罗马数字。

【例 1.14】

下面使用<ol>标记创建一个网站导航列表，代码如下：

```
<ol >
  <li><a href="#">和你在一起</a></li>
  <li><a href="#">白色百合花</a></li>
  <li><a href="#">似水年华</a></li>
  <li><a href="#">花蕊</a></li>
  <li><a href="#">美酒文学社</a></li>
  <li><a href="#">水木清华论坛</a></li>
</ol>
```

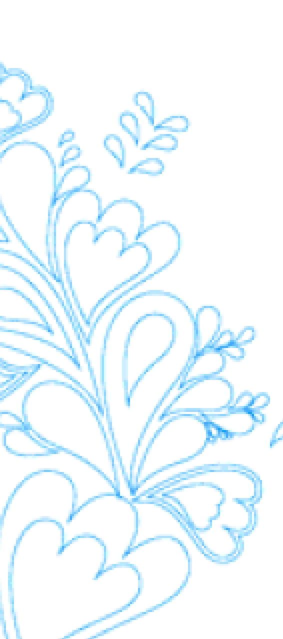
标记默认会对所有 li 列表项使用数字编号，运行效果如图 1-11 所示。现在为<ol>标记添加“type=“A” start=“2””代码，使用字母并且从第 2 项开始编号，再次运行，效果如图 1-12 所示。



图 1-11　使用数字编号的效果



图 1-12　使用字母编号的效果



1.4.2 项目符号列表

项目符号列表使用<ul>标记，其中每一个项目都使用<li>标记。项目符号标记的语法如下：

```
<ul>
    <li>项目符号</li>
    <li>项目符号</li>
    <li>项目符号</li>
    ...
</ul>
```

项目符号列表<ul>标记仅有一个 type 属性，type 属性可取的值为 circle(空心圆点)、disc(实心圆点)和 square(实心正方形)。

【例 1.15】

下面使用<ul>标记创建一个网站导航列表，并将 type 属性设置为 circle，代码如下：

```
<ul type="circle" >
   <li><a href="#">和你在一起</a></li>
   <li><a href="#">白色百合花</a></li>
   <li><a href="#">似水年华</a></li>
   <li><a href="#">花蕊</a></li>
   <li><a href="#">美酒文学社</a></li>
   <li><a href="#">水木清华论坛</a></li>
</ul>
```

运行后会看到一个使用空心圆点的列表，如图 1-13 所示。如果将 type 设置为 disc，则运行效果如图 1-14 所示。

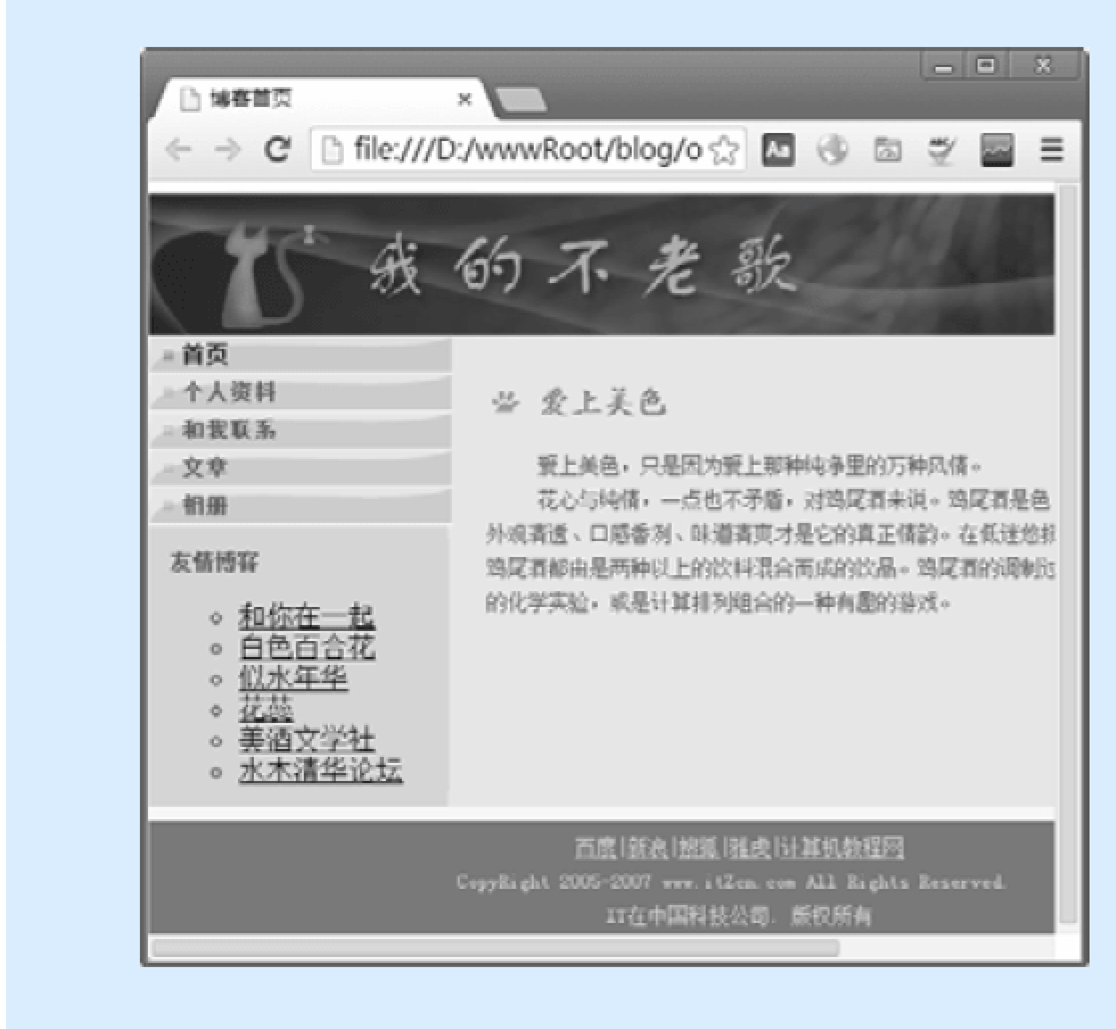


图 1-13　使用空心圆点的效果



图 1-14　使用实心圆点的效果

1.4.3 说明项目列表

说明项目列表可以用来给每一个列表项再加上一段说明性文字，说明性文字独立于列

表项另起一行显示。在应用中，列表项使用<dt>标记表示，说明性文字使用<dd>标记表示。

定义性列表的语法结构为：

```
<dl>
    <dt>第一项</dt>
    <dd>叙述第一项的定义</dd>
    <dt>第二项</dt>
    <dd>叙述第二项的定义</dd>
    ...
</dl>
```

【例 1.16】

演示说明性项目列表的使用：

```
<h2>香水品牌</h2>
<dl>
  <dt>Bijan</dt>
  <dd>毕扬(Bijan)由名牌服装设计师毕扬调制，最昂贵的香水，有浓郁而神秘的东方香味。 </dd>
  <dt>Chanel No.5</dt>
  <dd>香奈尔 5 号香水，其开瓶香味为花香乙醛调，持续香味为木香调，5 号香水的花香精致地注
释了女性独特的妩媚与婉约。 </dd>
 </dl>
```

执行效果如图 1-15 所示。



图 1-15　使用说明性项目列表的效果

1.5　表 格 标 记

美工或开发人员在设计网页的时候，为了使页面更加美观，通常会使用到表格。表格是 HTML 中一项非常重要的功能，利用其多种属性能够设计出多样化的表格，可以说，表格是网页排版的灵魂。下面详细介绍表格的结构、表格标记、表格属性及其应用。

1.5.1　表格的结构

表格主要由行和列(也叫单元格)组成。其中表格由<table>标记表示，行由<tr>标记表示，列由<td>标记表示，行和列都需要放在<table>和</table>之间。另外，还可以通过

<caption>标记定义表格的标题，<th>标记表示表格的表头。创建表格的基本语法结构如下：

```
<table>
    <caption> 表格标题</caption>
    <tr>
        <th>表头名称</th>...
    </tr>
    <tr>
        <td>文字内容</td><td>文字内容</td>...
    </tr>
</table>
```

上述基本结构中，表格的表示以行为单位，在行中包含列。其中一个<tr></tr>表示一行；一个<td></td>标记表示一列；<th></th>定义表头，一般可以不用。

【例 1.17】

创建一个 3 行 3 列的表格，要求定义表格标题，并将第一行设置为表头。所设计的表格的最终代码如下：

```
<h2>香水品牌</h2>
<table border="1">
  <caption>著名香水品牌</caption>
  <tr>
    <th>英文名称</th> <th>中文名称</th> <th>备注</th>
  </tr>
  <tr>
    <td>Chanel No.5</td> <td>香奈尔 5 号香水</td> <td>香奈尔</td>
  </tr>
  <tr>
    <td>LANCOME so magic</td> <td>兰蔻奇迹香水</td> <td>LANCOME 兰蔻</td>
  </tr>
</table>
```

表格的运行效果如图 1-16 所示。



图 1-16　表格的运行效果

1.5.2　表格的属性

上节的示例中，为了更加清楚地显示表格的结构，为表格设置了 border 属性。border 属性用于设置表格边框的宽度。除了 border 属性外，表格还具有许多其他属性。

1. 表格外边框的控制属性

表格的外边框控制属性共有 border、cellspacing 和 cellpadding 这 3 个，属性的作用说明如表 1-4 所示。

表 1-4　表格外边框的控制属性

属　性	说　明
border	控制表格边框的宽度。属性值为一整数，单位为像素
cellspacing	控制单元格边框到表格边框的距离，单位为像素
cellpadding	控制单元格内文字到单元格边框的距离，单位为像素

【例 1.18】

在例 1.17 的基础上为表格添加 border、cellpadding、cellspacing 属性，控制表格边框、单元格及文本字到单元格之间的大小。最终代码如下：

```
<table border="5" cellspacing="15" cellpadding="20">
  <caption>著名香水品牌</caption>
  <tr>
    <th>英文名称</th>
    <th>中文名称</th>
    <th>备注</th>
  </tr>
  <tr>
    <td>Chanel No.5</td> <td>香奈尔 5 号香水</td> <td>香奈尔</td>
  </tr>
  <tr>
    <td>LANCOME so magic</td> <td>兰蔻奇迹香水</td> <td>兰蔻</td>
  </tr>
</table>
```

执行结果如图 1-17 所示。

2. 表格的属性

一般情况下，表格的总长度和总宽度是根据各行和各列的内容的总和自动调整的。如果想要直接固定表格的大小，可以使用表格的 width 和 height 属性。width 和 height 属性分别用于控制表格的宽度和高度。除此之外，还可以控制表格的背景色和水平对齐方式。这几个属性的说明如表 1-5 所示。

表 1-5　表格的属性

属　性	说　明
width	控制表格的宽度。如果其取值为一整数，则单位为像素。若设置值为 n%，则表示表格的宽度为整个网页宽度的百分之 n
height	控制表格的高度。取值与 width 相同
bgcolor	设置表格的背景颜色
align	用于控制整个表格在网页水平面方向的对齐方式(left、right、center)

【例 1.19】

把例 1.18 中的表格宽度设置为百分之百，背景颜色设置为#CCCCCC，即添加 width="100%" bgcolor="#CCCCCC"代码。运行后的效果如图 1-18 所示。



图 1-17　设置间隔　　　　图 1-18　设置宽度和背景色

3. 行和列的属性

表格行的属性可以控制本行的高度、宽度、外框颜色、背景颜色、水平和垂直对齐方式。表格行的各属性如表 1-6 所示。

表 1-6　表格行的属性

属性名称	说　明
height	在<tr>标记中时，可以控制表格内某行的高度
bordercolor	控制表格中某行的外框颜色
bgcolor	控制该行单元格的背景颜色
align	控制本行中各单元格的内容在水平方向的对齐方式
valign	控制本行中各单元格的内容在垂直方向的对齐方式。取值为 top(上对齐)、middle(居中对齐)和 bottom(下对齐)

4. 单元格的属性

单元格属性用于控制表格中具体某个单元格的显示方式，单元格的属性如表 1-7 所示。

表 1-7　单元格的属性

属性名称	说　明
bordercolor	单元格的边框颜色
bgcolor	单元格的背景颜色
align、valign	单元格内的文字水平、垂直对齐方式
colspan	控制单元格合并右方的单元格数，达到延伸单元格的效果
rowspan	控制单元格合并下方的单元格数，达到垂直延伸单元格的效果

【例 1.20】

上述属性中的 bordercolor、bgcolor、align 和 valign 属性与表格和表格行相同。这里主要介绍使用 colspan 和 rowspan 属性实现单元格的合并。

这里以图 1-19 中的表格为例。



图 1-19　跨行和跨列表格

实现跨行和跨列的效果需要使用单元格的 rowspan 属性和 colspan 属性，其属性值为要跨越的行数和列数。

具体代码如下：

```
<table border="5" cellpadding="10" width="100%" bgcolor="#CCCCCC">
  <tr>
    <th width="25%">香水系列</th>
    <th width="50%">说明</th>
    <th width="25%">品牌</th>
  </tr>
  <tr>
    <td valign="top">奇迹香水</td>
    <td align="center">香调：水果花香调<br />
      前味：荔枝、小菖兰<br />
      中味：木兰花、青姜<br />
      后味：茉莉、麝香、琥珀</td>
    <td rowspan="2">兰蔻</td>
  </tr>
  <tr>
    <td valign="bottom">魔幻香水</td>
    <td>香调：清新花香调<br />
      前味：紫罗兰花瓣、胡椒粉<br />
      中味：保加利亚玫瑰<br />
      后味：香柏、三叶草花瓣</td>
  </tr>
  <tr>
    <td>爱恋香水</td>
    <td colspan="2" align="center">暂无资料</td>
  </tr>
</table>
```

1.6 表 单 标 记

表单在 Web 网页中用来给访问者填写信息，从而能获得用户信息，使网页具有交互的功能。一般是将表单设计在一个 HTML 文档中，当用户填写完信息后做提交(submit)操作时，表单的内容就从客户端的浏览器传送到服务器上，经过服务器上的 ASP 或 CGI 等处理程序处理后，再将用户所需信息传送回客户端的浏览器上，这样网页就具有了交互性。

1.6.1 创建表单

表单在 HTML 中由<form>标记定义，它是 HTML 的一个重要组成部分，主要用于搜集不同类型的用户输入和进行数据传递。在 HTML 表单中包含了很多表单元素，通过它们允许用户单击、选择和输入信息。HTML 表单的创建语法如下所示：

```
<form name="form name" method="method" action="url" enctype="value"
  target="target" id="id">
    <!-- 此处放表单元素，这里省略  -->
</form>
```

在上述代码中，表单各个属性的说明如表 1-8 所示。

表 1-8　表单属性说明

属性名称	说　明
name	表单的名称
method	设置表单的提交方式，有 GET 和 POST 两种
action	指向处理该表单页面的 URL，可以是相对位置或者绝对位置
enctype	设置表单内容的编码方式
target	设置返回信息的显示方式，可选值有_blank、_parent、_self 和_top
id	表单的 ID 号

1.6.2 创建表单元素

创建表单完成后，用户可以向表单中添加其他的标记元素了。一般情况下，按照用户需要填写的方式，分为输入类元素和菜单列表类元素。此外，还有一种 textarea 元素。

1. 输入类元素

输入类的元素一般都是以<input>标记开始，它的常用语法如下：

```
<input type="元素类型" name="元素名称" />
```

上述语法中最重要的是属性 type，该属性确定了元素的类型，表 1-9 对 type 的常用属性值进行了说明。

表 1-9　type 属性的常用值

type 取值	说　明
text	普通的文字输入字段
password	密码输入框，用户向该框中输入内容时，不显示具体内容，而是以*代替
radio	单选按钮
checkbox	复选框
button/submit/reset	普通按钮/提交按钮/重置按钮
image	图形域，也叫图像提交按钮
hidden	隐藏域，并不会显示到页面上，只是将内容传递到服务器中
file	文件域

可以向页面中添加元素供用户注册时接受用户输入。例如，下面的代码分别用来接受用户向表单中添加用户名、密码、性别和个人头像：

```
<form action="# " name="register" method="post" target=" parent">
   用户名称：<input name="txtUserName" type="text" /><br /><br />
   用户密码：<input name="txtUserPass" type="password" /><br /><br />
   用户性别：<input id="rdoBoy" name="Sex" type="radio" />男
          <input id="rdoGirl" name="Sex" type="radio" />女<br /><br />
   个人头像：<input id="txtImage" type="file" />
</form>
```

上述代码中，由于用户的性别只能是“男”或“女”，因此选中其中一个单选按钮时，另一个单选按钮就不能选中。要实现这样的功能，可以同时将单选按钮的 name 属性的值设为相同的。

2. textarea

textarea 称为文本区域框，它使用户能添加多行文本内容，例如个人介绍、图书简介或留言内容都可以使用<textarea>标记。它的语法如下：

```
<textarea id="文本域 ID" name="文本域名称" value="默认文本" rows="行数"
 cols="列数"> </textarea>
```

例如，为页面添加一个行数为 5，列数为 40 的文本域。代码如下：

```
<textarea name="message" rows="5" cols="40"></textarea>
```

3. 菜单列表类元素

菜单列表常常将<select>与<option>标记结合起来，如果将<select>标记的 multiple 属性的值设置为 true，则会向页面显示一个列表项。菜单列表标记的语法如下：

```
<select name="下拉菜单名称">
   <option value="" selected="selected">选项显示内容</option>
   <option value="选项值">选项显示内容</option>
</select>
```

菜单列表项可包含多个属性，如 name、size、value 等，属性说明如下。

- name：菜单和列表项的名称。
- size：显示选项的数目。
- value：获取选中项的值。
- multiple：设置列表中的项目是否多选。
- selected：默认选择项。

【例 1.21】

下面分别显示菜单项和列表项，并设置它们的属性，实现步骤如下。

步骤01 在页面中使用<select>标记创建一个可多选的列表，并指定宽度、大小和名称。代码如下：

```
<p class="text"> 喜欢的香水产地：
<select id="ad" name="ad" multiple="multiple" size="4" width="150">
   <option value="中国">中国</option>
   <option value="法国">法国</option>
   <option value="英国">英国</option>
   <option value="美国">美国</option>
   <option value="丹麦">丹麦</option>
   <option value="其他国家">其他国家</option>
</select></p>
```

步骤02 继续向页面中添加<select>和<option>标记，实现一个单选的下拉列表。代码如下：

```
<p class="text">喜欢的品牌：
<select id="band" name="band" style="width: 150px">
    <option value="香奈尔">香奈尔</option>
    <option value="兰蔻" selected="selected">兰蔻</option>
    <option value="兰黛">兰黛</option>
    <option value="雅顿">雅顿</option>
    <option value="其他">其他</option>
</select></p>
```

步骤03 运行页面查看效果，如图 1-20 所示。



图 1-20 创建菜单项和列表项

提示：截止到目前，本章已经向读者介绍了 HTML 中经常使用的一些标记元素，HTML 中的标记远远不止这些，其他的还有<span>、<fieldset>和<button>等，读者可以在网上找资料，也可以参考与 HTML 有关的书籍。

1.7　实战——制作卡通类页面

在本节之前已经介绍了 HTML 文档的结构、常用基础标记，以及在页面中使用列表、表格和表单的方法。在本章最后将综合这些知识制作一个卡通类的 HTML 完整页面，最终运行效果如图 1-21 所示。



图 1-21　卡通类页面的最终效果

如下所示为主要的实现步骤。

步骤 01　新建一个 HTML 文档结构，并将文件名保存为 index.html。

步骤 02　使用<title>标记设置页面标题为“青少年园地”。

步骤 03　使用<meta>标记设置页面使用 utf-8 编码。

步骤 04　向页面中插入一张图片，并使用<center>标记使其居中显示。代码如下：

```
<center>
  <img src="images/head jinhua.jpg" width="778" height="177" />
</center>
```

步骤 05　在图片下方添加一个一行两列的表格。设置表格为无边框和间隔、居中对齐、宽度为 778。

步骤 06　设置第 1 个单元格为水平居中对齐、垂直顶部对齐，并向单元格内插入一张图片。设置第 2 个单元格为垂直顶部对齐，背景颜色为#FFFFFF。此时的表格代码如下：

```
<table width="778" border="0" align="center" cellpadding="0"
  cellspacing="0">
    <tr>
```

```
      <td align="center" valign="top">
         <img src="images/ny list 001.jpg" alt="" width="188"
           height="66" />
      </td>
      <td valign="top" bgcolor="#FFFFFF">
         <!-- 第 2 个单元格  -->
      </td>
   </tr>
</table>
```

步骤 07 向上面第 2 个单元格内嵌套一个两行一列的表格。设置表格为无边框和间隔、居左对齐、宽度为 100%。设置第一行的单元格为垂直顶部对齐，并在单元格内插入 4 张图片。代码如下所示：

```
<table width="100%" border="0" align="left" cellpadding="0"
  cellspacing="0">
   <tr>
      <td valign="top">
         <img src="images/home.jpg" width="94" height="32" />
         <img src="images/bt1.jpg" />
         <img src="images/bt2.jpg" />
         <img src="images/bt3.jpg" />
      </td>
   </tr>
   <tr>
      <td valign="top"> <!-- 第 2 行的单元格  --></td>
   </tr>
</table>
```

步骤 08 在第 2 行的单元格内使用<h2>标记定义一个名为“早期教育”的标题。代码如下：

```
<h2>早期教育</h2>
```

步骤 09 使用<dl>标记罗列标题的中文名称和英文名称。代码如下：

```
<dl>
   <dt>中文名称</dt>
   <dd>早期教育</dd>
   <dt>英文名称</dt>
   <dd>Early Education</dd>
</dl>
```

步骤 10 创建一个段落，并使用
标记显示为两行。代码如下：

```
<p>    广义指从人出生到小学以前阶段的教育，狭义主要指上述
阶段的早期学习。<br/>
    家庭教育对早期教育有重大影响。其中“体能、智能、心理
能力三维平衡发展”理论最为科学，依据五万份孩子成长基准数据，提炼出孩子的九大成长目
标：安全感、意志力、目标感、注意力、记忆力、思维能力、平衡、力量和速度。 </p>
```

步骤 11 经过上面的步骤，表格的编辑与制作就结束了。在最外层表格的下方定义一个段落来显示页面的辅助信息。代码如下所示：

```
<p align="center">
```

```
<a href="#">关于我们</a> | <a href="#">免责声明</a>
 | <a href="#">广告合作</a> | <a href="#">联系方式</a>
<hr />
<center>
  版权所有 <b>窗内网</b>
</center>
</p>
```

上述代码使用<a>标记创建了 4 个链接，使用<hr/>标记绘制水平线，将链接信息与版权信息隔开。

1.8 本 章 习 题

1. 填空题

(1) 在一个 HTML 文档中使用__________标记页面的主体内容。

(2) 为网页添加描述信息时需要将元信息标记的__________属性指定为 description。

(3) HTML 中创建超链接时需要使用__________标记。

(4) 创建无序列表时，<ol>标记 type 属性的默认值是__________。

(5) 使用列表标记中的__________标记可以创建有序编号。

(6) 在定义列表时，可以使用<dl>标记的__________定义名称。

(7) 控制表格边框的属性为__________。

(8) 向表单添加文件上传文本框，需要将<input>标记的 type 属性指定为__________。

2. 选择题

(1) 标题标记符中__________标记表示最大的标题。

A. <h1>　　B. <h5>　　C. <h6>　　D. <h7>

(2) 下列使用 HTML 注释的情形正确的是__________。

A. <!-- 我是注释 -->　　B. //我是注释

C. /*我是注释*/　　D. 《我是注释》

(3) 阅读下面一段 HTML 代码，其显示效果为__________:

```
<body bgcolor="#FFFFFF" text="#FF0000">
<h1 color="#00FF00">网页设计</h1>
</body>
```

A. 白底红字　　B. 白底绿字　　C. 黑底绿字　　D. 黑底红字

(4) 在 HTML 中，定义有序列表的有序列表标记符是__________。

A. 　　B. 　　C. <dl>　　D.

(5) 关于 align 属性的取值，下列哪项是错误的？ __________

A. left、right、center　　B. right、left、middle

C. center、right、middle　　D. left、middle、center

(6) 在 HTML 中创建一个普通的表格时不应包括__________标记符。

A. <table>　　B. <title>　　C. <tr>　　D. <td>和<th>

(7) 定义表格中一行的标记是__________。

A. <tr>　　B. <th>　　C. <td>　　D. <table>

(8) 关于表格标记的说法，下面选项__________是不正确的。

A. <thead>、<tbody>和<tfoot>标记可以用来明确表格结构，它们分别对表首、表主体和表尾的样式进行设置。

B. width 和 height 属性只能在表格的<table>标记中使用，<td>和<tr>标记中不能使用。

C. 向表格中添加<caption></caption>并在它们的中间添加文本内容，所添加的文本内容会作为表格的标题。

D. 不仅可以通过表格的 background 属性添加背景，还可以通过在表格的 style 属性中设置参数的属性和值来添加背景。

3. 上机练习

(1) 制作一个表格

根据本章学习的表格知识，在 HTML 中绘制一个如图 1-22 所示的表格，注意单元的对齐方式以及单元格的合并。

(2) 制作一个表单

根据本章学习的表单知识，在 HTML 中绘制一个如图 1-23 所示的表单，注意表单元素属性的定义。



图 1-22　表格效果



图 1-23　表单效果



第2章

CSS 基 础

CSS 样式表用于为 HTML 标记设计样式。HTML 标记原本被设计为用于定义文档内容。通过使用<h1>、<p>、<table>这样的标记，表达“这是标题”、“这是段落”、“这是表格”之类的信息。同时文档布局由浏览器来完成，而不使用任何格式化标记。

由于浏览器不断地将新的 HTML 标记和属性(比如字体标记和颜色属性)添加到 HTML 规范中，创建文档内容清晰地独立于文档表现层的站点变得越来越困难。

为了解决这个问题，万维网联盟(W3C)这个非营利的标准化联盟，肩负起了 HTML 标准化的使命，并在 HTML 之外创造出样式(Style)。目前所有的主流浏览器均支持层叠样式表。本章将详细介绍样式表的由来和基础语法。

本章学习目标：

- 掌握 CSS 与 HTML 的结合方法
- 了解 CSS 的语法规则
- 理解 CSS 选择器的分组和继承
- 掌握 CSS 背景样式的使用
- 掌握 CSS 文本样式的使用
- 掌握 CSS 字体样式的使用
- 掌握 CSS 链接样式的使用
- 掌握 CSS 列表样式的使用
- 掌握表格和轮廓的样式使用
- 掌握框模型的样式使用

2.1 CSS 概述

CSS(Cascading Style Sheets)层叠样式表定义如何显示 HTML 元素，包括对字体、颜色、边框等样式进行设置。

2.1.1 CSS简介

CSS 样式通常保存在外部的.css 文件中。通过仅仅编辑一个简单的 CSS 文档，外部样式表可以同时改变站点中所有页面的布局和外观。

由于允许同时控制多重页面的样式和布局，CSS 可以称得上 Web 设计领域的一个突破。除此之外，还能够为每个 HTML 元素定义样式，并将其应用于人们所希望的任意多的页面中。如需进行全局的更新，只需简单地改变样式，然后网站中的所有元素均会自动地更新。

样式表从 1990 年代初 HTML 被发明开始，就以各种形式出现了，不同的浏览器结合了它们各自的样式语言，读者可以使用这些样式语言来调节网页的显示方式。

一开始，样式表是给读者用的，最初的 HTML 版本只含有很少的显示属性，由读者来决定网页应该怎样被显示。

但随着 HTML 的成长，为了满足设计师的要求，HTML 获得了很多显示功能。随着这些功能的增加，外来定义样式的语言越来越没有意义了。

1994 年哈坤・利提出了 CSS 的最初建议。伯特・波斯(Bert Bos)当时正在设计一个叫作 Argo 的浏览器，他们决定一起合作设计 CSS。

当时已经有过一些样式表语言的建议了，但 CSS 是第一个含有“层叠”概念的。在 CSS 中，一个文件的样式可以从其他的样式表中继承下来。读者在有些地方可以使用自己更喜欢的样式，在其他地方则继承，或“层叠”作者的样式。这种层叠的方式使作者和读者都可以灵活地加入自己的设计，混合各人的爱好。

哈坤于 1994 年在芝加哥的一次会议上第一次展示了 CSS 的建议，1995 年他与波斯一起再次展示这个建议。当时 W3C 刚刚创建，W3C 对 CSS 的发展很感兴趣，还为此组织了一次讨论会。哈坤、波斯和其他一些人(比如微软的托马斯・雷尔登)是这个项目的主要技术负责人。1996 年底，CSS 已经完成。1996 年 12 月 CSS 的第一个版本被推出。

1997 年初，W3C 内组织了专门管理 CSS 的工作组，其负责人是克里斯・里雷。这个工作组开始讨论第一版中没有涉及到的问题，其结果是 1998 年 5 月推出了第二版要求。

作为一项 W3C 推荐，CSS 2 发布于 1999 年 1 月 11 日。CSS 2 添加了对媒介(打印机和听觉设备)和可下载字体的支持。

CSS 不同的版本所支持的样式不同，表现如下。

(1) 1996 年 12 月发表的 CSS 1 的要求如下所示：

- 支持字体的大小、字形、强调。
- 支持字的颜色、背景的颜色和其他元素。

- 支持文章特征，如字母、词和行之间的距离。
- 支持文字的排列、图像、表格和其他元素。
- 支持边缘、围框和其他关于排版的元素。
- 支持 id 和 class。

(2) 1998 年 5 月 W3C 发表了 CSS 2，其中包括新的内容，如下所示：

- 绝对的、相对的和固定的定比特素、媒体型的概念。
- 双向文件和一个新的字体。

(3) CSS 2.1 修改了 CSS 2 中的一些错误，删除了其中基本不被支持的内容和增加了一些已有的浏览器的扩展内容。

(4) CSS 目前最新版本为 CSS 3，是能够真正做到网页表现与内容分离的一种样式设计语言。

相对于传统 HTML 的表现而言，CSS 能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力，并能够进行初步交互设计，是目前基于文本展示最优秀的表现设计语言。CSS 能够根据不同使用者的理解能力，简化或者优化写法，针对各类人群，有较强的易读性。

2.1.2 CSS的使用

CSS 语句运行在客户端，需要嵌入在 HTML 中借助浏览器来执行。CSS 语句可以以语句的形式直接嵌入 HTML 内部；也可以在 HTML 中引用外部 CSS 文件。CSS 语句所存在的位置有以下 3 种：

- 外部样式表。
- 内部样式表(位于<head>标记内部)。
- 内联样式(在 HTML 元素内部)。

上述 3 种样式存在形式可同时使用，也可单独使用，这种使用方式称作多重样式。

多重样式允许在单个 HTML 元素中、在 HTML 页的头元素中，或在一个外部的 CSS 文件中，甚至可以在同一个 HTML 文档内部引用多个外部样式表。

但多种样式的使用可能存在样式设置的冲突，如页面的大多表格背景色都是黄色，但其中一个背景色需要是红色，那么可以定义页面的表格颜色为黄色，并对指定的表格在其内部设置背景色为红色。

在这 3 种样式使用中，内联样式拥有最高的优先权，其次是内部样式表，最后是外部样式表。

这意味着内联样式将优先于以下的样式声明：<head>标记中的样式声明、外部样式表中的样式声明，或者浏览器中的样式声明(默认值)。

这几种方法中每一种方法都有其优缺点，可以根据以下几种情况，判断需要使用哪种样式设置：

- 当要在站点上所有或部分网页上一致地应用相同样式时，可使用外部样式表。在一个或多个外部样式表中定义样式，并将它们链接到所有网页，便能确保所有网页外观的一致性。如果决定更改样式，只需在外部样式表中修改一次，而该更改

会反映到所有与该样式表相链接的网页上。

- 当只是要定义当前网页的样式时，可使用嵌入的样式表。嵌入的样式表是一种级联样式表，嵌在网页的<head>标记符内。嵌入的样式表中的样式只能在同一网页上使用。
- 当只需要定义当前 html 标记的样式时，可使用内联样式。内联样式设置在 html 标记内部，只对一个 html 标记有效。

1. 外部样式表

样式通常保存在外部的.css 文件中。通过仅仅编辑一个简单的 CSS 文档，外部样式表使你有能力同时改变站点中所有页面的布局和外观。

外部样式表通常以.css 作为文件扩展名，例如 Mystyles.css。然后在需要此样式的页面中将其链接进来，代码如下：

```
<link href="Mystyles.css" rel="stylesheet" type="text/css"/>
```

2. 内部样式表

内部样式表通常放在<head>标记内，内部样式表以<style type="text/css">标记开始，以</style>标记结束，在这两个标记之间编辑样式代码，如下所示：

```
<style type="text/css">
    /*样式设置*/
</style>
```

3. 内联样式

内联样式放在 HTML 标记内部，以标记属性的方式为标记设置样式。该属性为 style 属性，如为<div>标记设置样式，使其背景色为黄色，代码如下：

```
<div style="background-color: #FFFF00">
内联样式
</div>
```

【例 2.1】

分别对一个页面的<div>标记定义外部样式表、内部样式表和内联样式表，为页面添加 3 个<div>标记，查看其样式表的执行效果，步骤如下。

步骤 01 定义一个外部的样式表 style.css 并添加代码如下：

```
div {
    background-color: #FFFF00;
    text-align: center;
}
```

上述代码表示将背景色设置为黄色，并将字体居中显示。背景色默认是白色的，而字体默认是靠左对齐。

步骤 02 创建页面，并在<head>标记下添加外部样式表，同时在页面的<body>标记下添加<div>代码。

具体如下：

```
<!DOCTYPE html>

<html>
<head>
    <title></title>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div>
        外部样式
    </div>
</body>
</html>
```

上述代码中没有定义<div>的样式，只是添加了文字“外部样式”。

步骤 03　向页面中添加<div>并添加其内部样式，即直接对<div>标记进行样式设置，设置其背景色为蓝色，而不定义字体的对齐方式，代码如下：

```
<div style="background-color: #00FFFF">
    内部样式
</div>
```

步骤 04　在<head>标记下添加内部样式表 div1，该样式表只定义背景色为红色，不定义字体的对齐方式，代码如下：

```
<style type="text/css">
    .div1 {
        background-color: #FF00FF;
    }
</style>
```

步骤 05　在页面中添加<div>，使用内部的样式，代码如下：

```
<div class="div1">
    内联样式
</div>
```

步骤 06　在浏览器中运行该页面，其效果如图 2-1 所示。虽然这 3 个<div>块使用了不同的样式，但字体都是居中显示的，因此可见这 3 种样式仅仅是相互冲突的部分遵循优先顺序，而不冲突的部分根据已经定义的样式来显示。

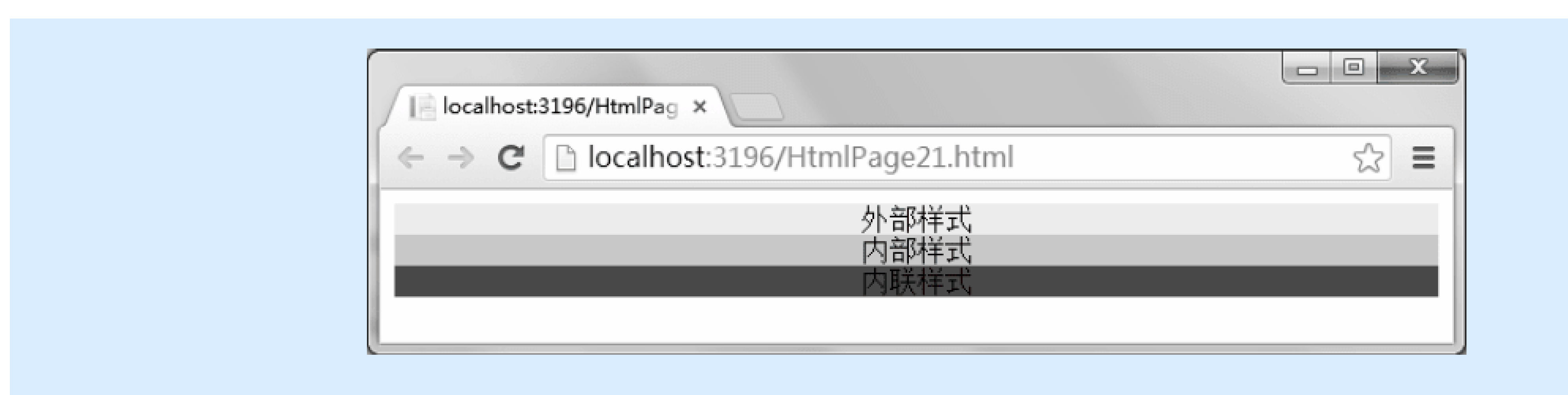


图 2-1　样式的使用

2.2 CSS 基础语法

本章 2.1 节介绍了 CSS 代码的使用方式，本节介绍 CSS 代码的基础语法。CSS 代码的书写是针对具体的 HTML 标记，其定义方式是样式属性名称、冒号、属性值和分号，如下所示：

```
属性名：属性值;
```

上述代码是建立在指定的 HTML 元素的基础上，如例 2.1 的步骤 01 是建立在 div 的基础上，有着背景色和对齐方式这两个属性的设置，分别设置其背景色为黄色；设置其对齐方式为居中显示，代码如下：

```
div {
    background-color: #FFFF00;
    text-align:center;
}
```

上述代码中，div 是 HTML 中的元素，为元素设置样式，需要在元素后面添加大括号“{}”，其样式设置放在大括号内部。上述代码定义之后，该样式设置将作用于引用该样式文件的页面的所有 div 元素。

在 CSS 中，除了为元素设置样式，还可以定义选择器来设置样式，再在 HTML 元素中根据选择器来显示样式。上述代码可以看作是 div 选择器样式，该样式下有两条样式设置语句(样式声明)。

例 2.1 中的步骤 04 为 div 元素设置了名为“div1”的选择器，并在 div 元素中通过 class 属性进行引用。

选择器通常是需要改变样式的 HTML 元素，属性(property)是希望设置的样式属性(style attribute)。每个属性有一个值。属性和值被冒号分开。

如果要定义不止一个样式声明，则需要用分号将每个声明分开。下面的例子展示出如何定义一个红色文字的居中段落。最后一条规则是不需要加分号的，因为分号在英语中是一个分隔符号，不是结束符号。然而，大多数有经验的设计师会在每条声明的末尾都加上分号，这么做的好处是，当从现有的规则中增减声明时，会尽可能地减少出错的可能性。

是否包含空格不会影响 CSS 在浏览器的工作效果，同样，与 XHTML 不同，CSS 对大小写不敏感。不过存在一个例外：如果涉及到与 HTML 文档一起工作的话，class 和 id 名称对大小写是敏感的。

一些特殊样式的属性值是有着特殊格式的，如颜色可以是指定的单词，也可以是十六进制的颜色值或颜色 RGB 值。如为段落元素 P 设置红色字体，有如下几种方式。

(1) 使用英文单词 red：

```
p { color: red; }
```

(2) 使用十六进制的颜色值：

```
p { color: #ff0000; }
```

(3) 使用 CSS 的缩写形式：

```
p { color: #f00; }
```

(4) 使用颜色 RGB 值“红绿蓝”：

```
p { color: rgb(255,0,0); }
p { color: rgb(100%,0%,0%); }
```

注意：当使用 RGB 百分比时，即使当值为 0 时，也要写百分比符号。但是在其他的情况下就不需要这么做了。比如说，当尺寸为 0 像素时，0 之后不需要使用 px 单位，因为 0 就是 0，无论单位是什么。

2.3 CSS 高级语法

CSS 高级语法是对选择器的升级，包括选择器分组、继承和重写等。通过对选择器进行操作，影响页面中元素的显示样式。

1. 选择器的分组

对选择器进行分组可以使被分组的选择器分享相同的声明。用逗号将需要分组的选择器分开。

【例 2.2】

对标题元素进行选择器分组，使 h1、h2、h3 都显示红色字体，创建页面并添加 h1、h2、h3 和一个段落，查看其显示效果，代码如下：

```
<head>
    <style type="text/css">
        h1, h2, h3 {
            color: red;
        }
    </style>
</head>

<body>
    <h1>标题 h1</h1>
    <h2>标题 h2</h2>
    <h3>标题 h3</h3>
    <p>段落</p>
</body>
```

在浏览器中运行上述代码，其效果如图 2-2 所示。

在图 2-2 中可见，h1、h2、h3 有着各自不同的默认字体大小，对其进行了颜色设置之后，这 3 个标记有了相同的字体颜色设置，但仍以不同的字体大小显示。段落没有进行样式设置，显示的是默认样式。



图 2-2　选择器分组

2. 继承

在 CSS 中，子元素可以从父元素中继承属性。如对于 body 元素中的 div 子元素，可以继承 body 元素的样式设置。

如果子元素需要使用不同的样式，那么可对子元素进行单独设置。元素样式的设置也有着子元素和父元素不同的优先级，子元素的样式优先级大于父元素，如例 2.3 所示。

【例 2.3】

对 body 元素中的 div 子元素设置样式为居中红色字体；在 div 元素中段落样式为靠左对齐黑色字体；向 div 中添加标题和段落，查看显示效果。代码如下：

```
<head>
    <style type="text/css">
        div {
            color: red;
            text-align: center;
        }
        p {
            color: black;
            text-align: left;
        }
    </style>
</head>
<body>
    <div>
        <h1>白居易</h1>
        <p>
            白居易(772~846)，字乐天，号香山居士，又号醉吟先生，河南新郑(今河南郑州新郑市)人， 唐代伟大的现实主义诗人，唐代三大诗人之一。白居易与元稹共同倡导新乐府运动，世称“元白”，与刘禹锡并称"刘白"。
        </p>
    </div>
</body>
```

上述代码中定义了 div 和 p 的样式，并没有定义标题的样式。但 h1 元素在 div 的内部，是 div 元素的子元素，因此遵循 div 样式。

虽然 p 元素也是 div 元素的子元素，但代码中有对 p 元素样式的单独定义，从优先级来看，p 元素显示其单独定义的样式，效果如图 2-3 所示。

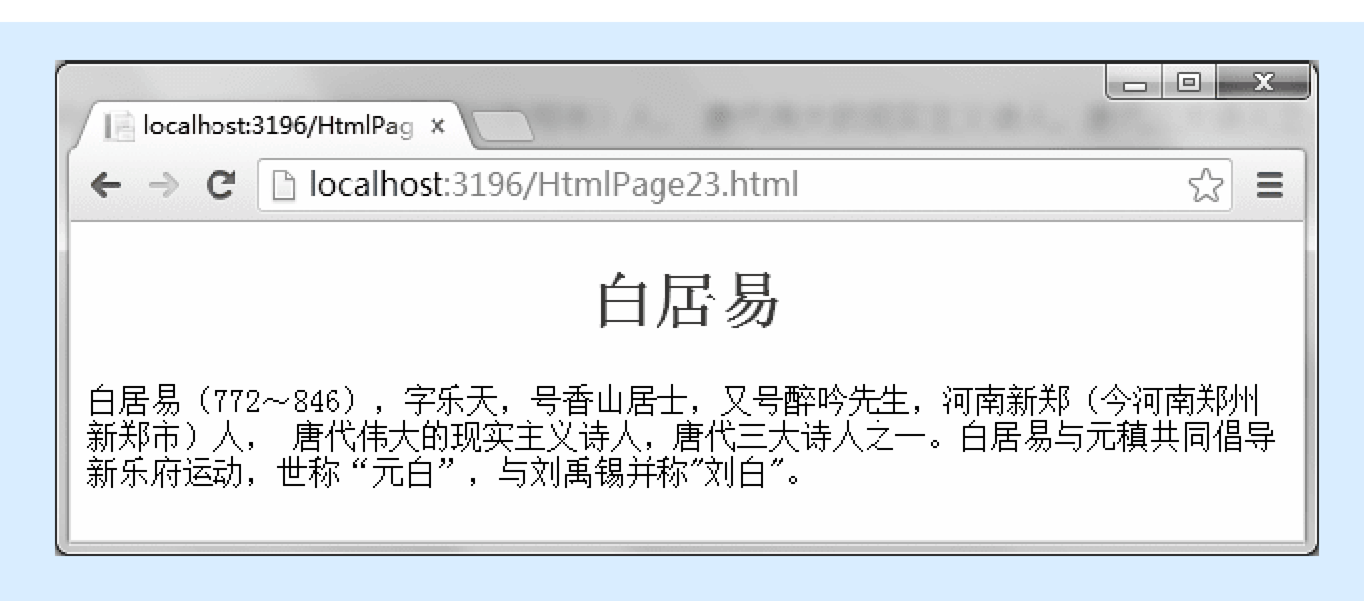


图 2-3　样式的继承和重写

注意：并不是所有的浏览器都支持样式的继承，例如，Netscape 4 就不支持继承，它不仅忽略继承，而且也忽略应用于 body 元素的规则。IE 浏览器中直到 IE 6 还存在相关的问题，在表格内的字体样式会被忽略。

3. 注释

CSS 中，语句是可以嵌入在 HTML 页面中的，但是其注释方式与 HTML 并不相同。CSS 代码放在<style type="text/css">和</style>标记之间，使用“/*”和“*/”符号作为注释语句的起始符号。如将 div 块中的字体颜色语句设置为注释语句，代码如下：

```
div {
    /*color: red;*/
    text-align: center;
}
```

2.4　CSS 的常用样式

本章前面几个小节总结了 CSS 样式的使用，本节将详细介绍 CSS 常用的样式及其样式设置。

2.4.1　CSS常用样式概述

浏览器中元素和数据的显示离不开样式的控制，而 CSS 语言是页面样式控制的主流。通常通过 style 属性来定义页面显示样式，其常用的样式及其属性控制类型如表 2-1 所示。

表 2-1　CSS 的常用属性

样　式	属　性
大小	font-size(x-large 特大；xx-small 极小，可用数值单位：PX、PD)
样式	font-style(oblique 偏斜体；italic 斜体；normal 正常)
行高	line-height(单位：PX、PD、EM)

续表

样　式	属　性
粗细	font-weight(bold 粗体；lighter 细体；normal 正常)
修饰	text-decoration(underline 下划线；overline 上划线；line-through 删除线；blink 闪烁)
常用字体	font-family
背景属性	background 色彩　background-color: #FFFFFF。 图片　background-image: url。 重复　background-repeat: no-repeat
字间距	letter-spacing
对齐	text-align(justify 两端对齐；left 左对齐；right 右对齐；center 居中)
缩进	text-indent
垂直对齐	vertical-align(top；text-top；middle；bottom；text-bottom)
空格	e-space(pre 保留；nowrap 不换行)
边框属性	Border border-style(dotted 点线；dashed 虚线；soliddouble 双线；groove 槽线；ridge 脊状；inset 凹陷；outset 凸现)。 border-width：边框宽度。 border-color：颜色
列表属性	List-style(list-style-type：disc 圆点；circle 圆圈；square 方块；decimal 数字 lower-roman 小罗马数字)

通过 CSS 样式表单，可以对 XML 元素的外观进行设置，如文本的样式、布局、背景颜色和边框等。

CSS 能够很好地控制输出的样式，比如色彩、字体、大小等，但是 CSS 只适合用于输出比较固定的最终文档。CSS 的优点是简洁，消耗系统资源少。

2.4.2　背景样式

背景样式的设置包括背景颜色设置、背景图片路径设置、背景图片重复设置、背景图片百分比设置、背景图片位置设置等。

1. 背景色

使用 background-color 属性为元素设置背景色。这个属性接受任何合法的颜色值。如果希望背景色从元素中的文本向外稍有延伸，只需增加一些内边距。

background-color 不能继承，其默认值是 transparent。transparent 有“透明”之意。也就是说，如果一个元素没有指定背景色，那么背景就是透明的，这样其父元素的背景才能可见。

2. 背景图像

要把图像放入背景，需要使用 background-image 属性。background-image 属性的默认值是 none，表示背景上没有放置任何图像。

如果需要设置一个背景图像，必须为这个属性设置一个 URL 值，如下所示：

```
body { background-image: url(img.gif); }
```

大多数背景都应用到 body 元素，不过并不仅限于此。在 CSS 中可以为元素设置背景图片，甚至可以向 textarea 和 select 等替换元素的背景应用图像，background-image 也不能继承。事实上，所有背景属性都不能继承。

3. 背景重复

由于背景图片有大有小，若图片大于元素区域，那么图片默认不完全显示；若图片小于元素区域，那么图片将在区域内默认重复显示。

如果需要在页面上对背景图像进行平铺，可以使用 background-repeat 属性。该属性的可取值如表 2-2 所示。

表 2-2　背景图片的重复属性

属 性 值	说　明
repeat	默认。背景图像将在垂直方向和水平方向重复
repeat-x	背景图像将在水平方向重复
repeat-y	背景图像将在垂直方向重复
no-repeat	背景图像将仅显示一次
inherit	规定应该从父元素继承 background-repeat 属性的设置

4. 背景定位

利用 background-position 属性改变图像在背景中的位置。为 background-position 属性提供值有很多种方法，一种是使用位置关键字，如 top、bottom、left、right 和 center；另一种是使用百分数值；还有一种是使用准确的像素来描述位置。具体如表 2-3 所示。

表 2-3　背景定位

表示方法	取　值	说　明
关键字	top left、top center、top right center left、center center、center right bottom left、bottom center、bottom right	如果仅使用一个关键词，那么第二个值默认为 center
百分数值	x% y%	第一个值是水平位置，第二个值是垂直位置。左上角是 0% 0%。右下角是 100% 100%。如果仅使用一个值，另一个值默认为 50%

续表

表示方法	取　值	说　明
位置描述	xpos ypos	第一个值是水平位置，第二个值是垂直位置。左上角是 0 0。单位是像素(0px 0px)或任何其他的 CSS 单位。如果仅使用一个值，另一个值默认为 50%。可以混合使用%和 position 值

5. 背景关联

如果文档比较长，那么当文档向下滚动时，背景图像也会随之滚动。当文档滚动到超过图像的位置时，图像就会消失。

可以通过 background-attachment 属性防止这种滚动。通过这个属性，可以声明图像相对于可视区是固定的(fixed)，因此不会受到滚动的影响。该属性的可取值及其说明如表 2-4 所示。

表 2-4　背景关联

取　值	说　明
scroll	随着页面的滚动背景图片将移动
fixed	随着页面的滚动背景图片不会移动
inherit	继承

background-attachment 属性的默认值是 scroll，也就是说，在默认情况下，背景会随文档滚动。

【例 2.4】

创建页面并添加表格元素，设置表格为一行三列的表格，为其 3 个单元格设置相同的背景图片，使用不同的背景设置，步骤如下。

步骤 01　定义 3 个单元格的样式，代码如下：

```
<style type="text/css">
   td {
      height: 200px;
      width: 200px;
      background-color: green;
   }
   .auto-style1 {
      background-image: url('Images/tutu.jpg');
   }
   .auto-style2 {
      background-image: url('Images/tutu.jpg');
      background-repeat: no-repeat;
      background-position: 50%;
   }
   .auto-style3 {
      background-image: url('Images/tutu.jpg');
```

```
        background-position: bottom right;
    }
</style>
```

步骤 02 向页面添加表格，并引用上述样式，代码如下：

```
<table>
    <tr>
        <td class="auto-style1"></td>
        <td class="auto-style2"></td>
        <td class="auto-style3"></td>
    </tr>
</table>
```

步骤 03 运行上述代码，其效果如图 2-4 所示。



图 2-4 使用背景样式

如图 2-4 所示，表格中的 3 个单元格使用了相同的区域大小和相同的背景图片，但显示效果不同。

- 第一个单元格使用默认设置，图片靠左上角显示，在超出背景图片的空间(右侧和下方)有图片的重复。
- 第二个单元格使用了不重复样式和图片居中样式，图片显示在该单元格的中间位置。超出图片的空间是空白的，其空白区域将显示该区域的背景色。
- 第三个单元格使用了图片靠右下显示，该单元格中左侧和上方有图片的重复。

2.4.3 文本样式

CSS 文本属性可定义文本的外观，包括文本的对齐方式、缩进、字间隔、字符间隔、字符转换和文本装饰等。

1. 缩进文本

把 Web 页面上的段落的第一行缩进，这是一种最常用的文本格式化效果。CSS 提供了 text-indent 属性，可以方便地实现文本缩进。

通过使用 text-indent 属性，所有元素的第一行都可以缩进一个给定的长度，甚至该长度可以是负值。其可能取值如下所示：

- 固定的缩进，默认值为 0。
- 百分比缩进，定义基于父元素宽度的百分比的缩进。
- inherit，规定从父元素继承 text-indent 属性的值。

text-indent 还可以设置为负值。利用这种技术，可以实现很多有趣的“悬挂缩进”，即第一行悬挂在元素中余下部分的左边。

不过，如果对一个段落设置了负值，那么首行的某些文本可能会超出浏览器窗口的左边界。

为了避免出现这种显示问题，建议针对负缩进再设置一个外边距或一些内边距。

注意：一般来说，可以为所有块级元素应用 text-indent，但无法将该属性应用于行内元素，图像之类的替换元素上也无法应用 text-indent 属性。如果一个块级元素(比如段落)的首行中有一个图像，它会随该行的其余文本移动。

2. 对齐方式

定义文本的对齐方式使用 text-align 属性，该属性定义文本的水平对齐方式，其默认值是 left(左对齐)。text-align 属性的取值包括左对齐、右对齐、居中对齐和两端对齐，具体如下所示。

- left：把文本排列到左边。
- right：把文本排列到右边。
- center：把文本排列到中间。
- justify：实现两端对齐文本的效果。
- inherit：规定从父元素继承 text-align 属性的值。

将块级元素或表元素居中，要通过在这些元素上适当地设置左、右外边距来实现。

值 justify 可以使文本的两端都对齐。在两端对齐文本中，文本行的左右两端都放在父元素的内边界上。然后，调整单词和字母间的间隔，使各行的长度恰好相等。

两端对齐文本在打印领域很常见。通常由用户浏览器(而不是 CSS)来确定两端对齐文本如何拉伸，以填满父元素左右边界之间的空间。

当前浏览器中，有些浏览器可能只在单词之间增加额外的空间，而另外一些浏览器可能会平均分布字母间的额外空间。还有一些用户浏览器可能会减少某些行的空间，使文本挤得更紧密。所有这些做法都会影响元素的外观，甚至改变其高度，这取决于用户浏览器的对齐选择影响了多少文本行。

CSS规范特别指出，如果letter-spacing属性(定义字母间隔)指定为一个长度值，那么用户浏览器不能进一步增加或减少字符间的空间。

CSS 中没有指定应当如何处理连字符(-)。大多数两端对齐文本都使用连字符将长单词分开放在两行上，从而缩小单词之间的间隔，改善文本行的外观。

不过，由于 CSS 没有定义连字符行为，所以用户浏览器不能自动添加连字符。因此，在 CSS 中，两端对齐文本看上去没有打印出来的好看，特别是元素可能太窄，以至于每行只能放下几个单词。

3. 字间隔

字间隔属性可以改变汉字或单词之间的间隔，其属性名称为 word-spacing。该属性有 3 种取值：normal、数值和 inherit。normal 是单词间的标准间隔，是该属性的默认值。

word-spacing 属性接受一个正长度值或负长度值。如果提供一个正长度值，那么字之间的间隔就会增加。为 word-spacing 设置一个负值，会把间隔拉近。

所有浏览器都支持 word-spacing 属性，但是任何版本的 IE(包括 IE 8)都不支持属性值 inherit。

CSS 把“字(word)”定义为任何非空白字符组成的串，并由某种空白字符包围。这个定义没有实际的语义，它只是假设一个文档包含由一个或多个空白字符包围的字。支持 CSS 的用户浏览器不一定能确定一个给定语言中哪些是合法的字，而哪些不是。

尽管这个定义没有多大价值，不过它意味着采用象形文字的语言或非罗马书写体往往无法指定字间隔。

4. 字符间隔

word-spacing 属性定义字间隔，而 letter-spacing 属性定义字符间隔。与 word-spacing 的区别在于，字符间隔修改的是字符或字母之间的间隔。

与 word-spacing 属性一样，letter-spacing 属性的可取值包括所有长度。默认值是 normal。输入的长度值会使字母之间的间隔增加或减少指定的量。

letter-spacing 属性的使用与 word-spacing 完全一样，只是针对的对象不同，其使用方式和可取的值参考 word-spacing 属性。

5. 字符转换

text-transform 属性处理文本的大小写。这个属性有 5 个可取的值，如下所示。

- none：默认。定义带有小写字母和大写字母的标准的文本。
- capitalize：文本中的每个单词以大写字母开头。
- uppercase：定义仅有大写字母。
- lowercase：定义无大写字母，仅有小写字母。
- inherit：规定从父元素继承 text-transform 属性的值。

默认值 none 对文本不做任何改动，将使用源文档中的原有大小写。顾名思义，uppercase 和 lowercase 将文本转换为全大写和全小写字符。最后，capitalize 只对每个单词的首字母大写。

作为一个属性，text-transform 可能无关紧要，不过如果要把所有 h1 元素变为大写，这个属性就很有用。不必单独地修改所有 h1 元素的内容，只需使用 text-transform 为你完成这个修改。

使用 text-transform 有两方面的好处。首先，只需写一个简单的规则来完成这个修改，而无须修改 h1 元素本身。其次，如果以后决定将所有大小写再切换为原来的大小写，可以更容易地完成修改。

6. 文本装饰

文本修饰属性 text-decoration 是一个有趣的属性，该属性提供了多个有趣的行为，其可取值如下所示。

- none：默认。定义标准的文本。
- underline：定义文本下的一条线。
- overline：定义文本上的一条线。
- line-through：定义穿过文本下的一条线。
- blink：定义闪烁的文本。
- inherit：规定从父元素继承 text-decoration 属性的值。

underline 会对元素加下划线，就像 HTML 中的 U 元素一样。overline 的作用恰好相反，会在文本的顶端画一个上划线。值 line-through 则在文本中间画一个贯穿线，等价于 HTML 中的 S 和 strike 元素。blink 会让文本闪烁，类似于 Netscape 支持的 blink 标记。


注意：IE、Chrome 和 Safari 不支持 blink 属性值。

none 值会关闭原本应用到一个元素上的所有装饰。通常，无装饰的文本是默认外观，但也不总是这样。

例如，链接默认地会有下划线。如果希望去掉超链接的下划线，可以显式地使用 text-decoration 属性去掉链接的下划线，还需要注意链接文字的颜色。链接文字的颜色通常与正文颜色不同。

如果使用两个不同的装饰来匹配同一元素，那么优先级高的属性值会完全取代另一个值。因为 text-decoration 值是替换而不是累积起来。

7. 处理空白符

white-space 属性会影响到用户对源文档中空格、换行和 Tab 字符的处理。通过使用该属性，可以影响浏览器处理字之间和文本行之间的空白符的方式。white-space 属性的可取值如下所示。

- normal：默认。空白会被浏览器忽略。
- pre：空白会被浏览器保留。其行为方式类似于 HTML 中的<pre>标记。
- nowrap：文本不会换行，文本会在同一行上继续，直到遇到
标记为止。
- pre-wrap：保留空白符序列，但是正常地进行换行。
- pre-line：合并空白符序列，但是保留换行符。
- inherit：规定应该从父元素继承 white-space 属性的值。

如果将 white-space 设置为 pre，受这个属性影响的元素中，空白符的处理就像 XHTML 的 pre 元素一样，空白符不会被忽略。

从某种程度上讲，默认的 XHTML 处理已经完成了空白符处理：把所有空白符合并为一个空格。文本在 Web 浏览器中显示时，各个字之间只会显示一个空格，同时忽略元素中的换行。

注意：经测试，IE7 以及更早版本的浏览器不支持 pre 属性值，因此应使用其他浏览器来验证该属性值的显示效果。

与 pre 属性值相对的值是 nowrap，它会防止元素中的文本换行，除非使用了一个 br 元素。在 CSS 中使用 nowrap 非常类似于 HTML4 中用<tdnowrap>将一个表单元格设置为不能换行，不过 white-space 值可以应用到任何元素。

CSS 2.1 引入了值 pre-wrap 和 pre-line，这在以前版本的 CSS 中是没有的。这些值的作用是允许创作人员更好地控制空白符处理。

如果元素的 white-space 设置为 pre-wrap，那么该元素中的文本会保留空白符序列，但是文本行会正常地换行。如果设置为这个值，源文本中的行分隔符以及生成的行分隔符也会保留。

pre-line 与 pre-wrap 相反，会像正常文本中一样合并空白符序列，但保留换行符。

8. 文本方向

如果阅读的是英文书籍，就会从左到右、从上到下地阅读，这就是英文的流方向。不过，并不是所有语言都如此。我们知道古汉语就是从右到左来阅读的，当然还包括希伯来语和阿拉伯语等。CSS 引入了 direction 属性来描述其方向性。

direction 属性影响块级元素中文本的书写方向、表中列布局的方向、内容水平填充其元素框的方向，以及两端对齐元素中最后一行的位置。

direction 属性有 3 个值，如下所示。

- ltr：默认。文本方向从左到右。
- rtl：文本方向从右到左。
- inherit：规定从父元素继承 direction 属性的值。

【例 2.5】

创建页面并添加表格元素，设置表格为两行三列的表格。其中首行为标题行，只有中间的单元格有文本“白居易”。第二行有 3 个单元格，分别添加白居易的 3 首诗词。设置这 4 个单元格的文本样式，查看显示效果，步骤如下。

步骤 01 首先设置单元格的样式，定义标题行字间距为 1em，并使字体居中显示，代码如下：

```
<style type="text/css">
    .td1 {
        letter-spacing: 1em;
        text-align: center;
    }
</style>
```

步骤 02 接着定义第二行左侧的单元格样式，定义其字体靠右显示，并使用 white-space 属性保持文本原样输出，代码如下：

```
.td2 {
    text-align: right;
```

```
    white-space: pre;
}
```

步骤 03 之后定义第二行中间的单元格样式，定义其字体居中显示，并使用 white-space 属性忽略空格、保留文本换行，同时添加文本的下划线，代码如下：

```
.td3 {
    text-align: center;
    white-space: pre-line;
    text-decoration: underline;
}
```

步骤 04 定义最后一个单元格样式，使其字体靠左显示，不定义其 white-space 属性，代码如下：

```
.td4 {
    text-align: left;
}
```

步骤 05 向页面中添加表格，引用上述样式。在添加白居易的诗词时，左侧单元格文本没有空格，只有换行；中间单元格中的文本有空格和换行；右侧单元格中的文本有空格和换行，代码如下：

```
<table>
    <tr>
        <td> </td>
        <td class="td1">白居易</td>
        <td> </td>
    </tr>
    <tr>
        <td class="td2">
            离离原上草，一岁一枯荣。
            野火烧不尽，春风吹又生。
            远芳侵古道，晴翠接荒城。
            又送王孙去，萋萋满别情。
        </td>
        <td class="td3">
            帝城春欲暮，喧喧车马度。
            共道牡丹时，相随买花去。
            贵贱无常价，酬直看花数：
            灼灼百朵红，戋戋五束素。
        </td>
        <td class="td4">
            江南好，风景旧曾谙。
            日出江花红胜火，春来江水绿如蓝，
            能不忆江南？
        </td>
    </tr>
</table>
```

上述代码的执行效果如图 2-5 所示。注意到中间单元格中的文本被去除了空格，右侧单元格中的文本根据单元格的宽度被换行，而不是根据代码中的换行而换行。

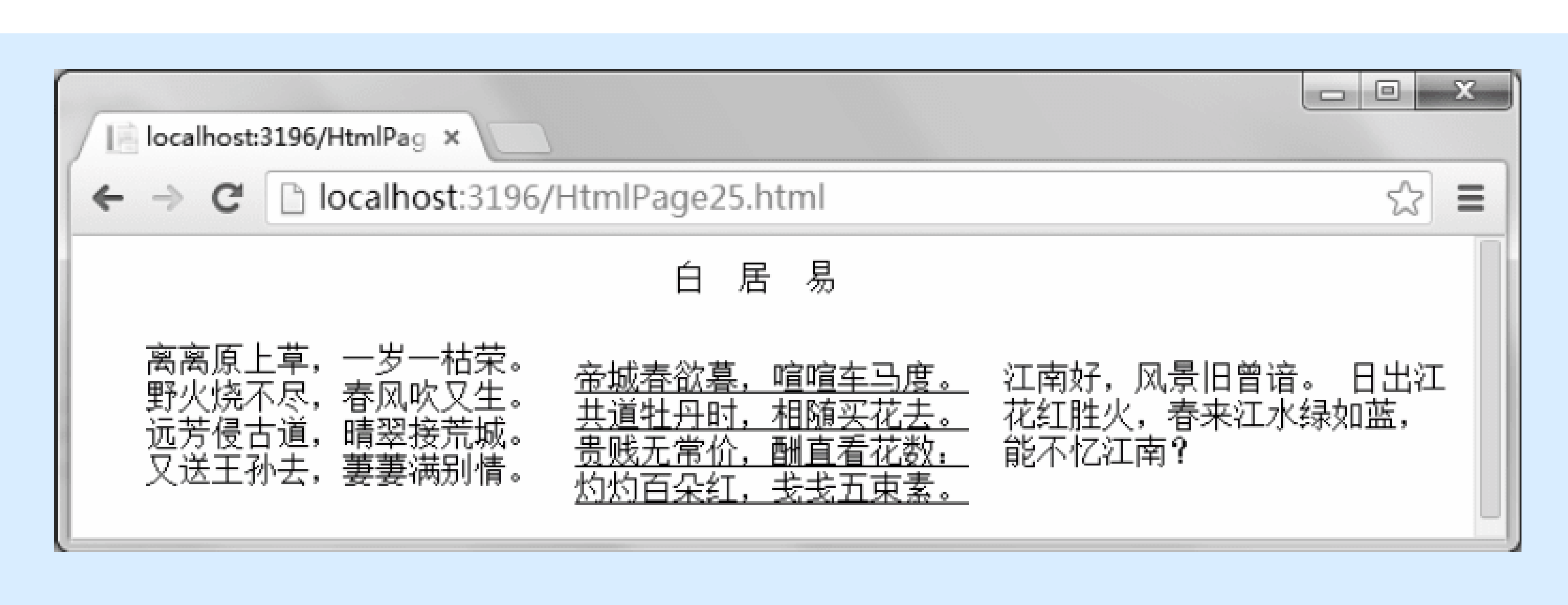


图 2-5　使用文本样式

2.4.4　字体样式

设置字体属性是样式表的最常见的用途之一。不过，尽管字体选择很重要，但是目前还没有一种办法能确保在 Web 上一致地使用字体，因为没有一种统一描述字体和字体变形的方法。在 CSS 中，有两种不同类型的字体系列，如下所示：

- 通用字体系列拥有相似外观的字体系统组合(比如 Serif 或 Monospace)。
- 特定字体系列具体的字体系列(比如 Times 或 Courier)。

除各种特定的字体系列外，CSS 还定义了 5 种通用字体系列：Serif 字体、Sans-serif 字体、Monospace 字体、Cursive 字体和 Fantasy 字体。

字体系列 Times、Times New Roman 和 TimesNR 可能很类似，甚至完全相同。Times New Roman 和 TimesNR 实际上是可以互换的。但是浏览器并不一定采用哪种字体系列，因此在使用字体样式的同时，需要设置字体系列。

与文字处理软件相比，CSS 对字体并没有提供更多的最终控制。比如我们知道加载一个 Microsoft Office 文档时，其显示字体取决于本机已经安装的字体。如果文档字体与本机所安装的字体不同，那么原有的字体将被改变。使用 CSS 设计的文档也是如此。

人们通常认为的 Times 是所有这些变形字体的一个组合。换句话说，Times 实际上是一个字体系列(Font Family)，而不只是单个的字体，尽管大多数人都认为这种字体就是某一种字体。

Times 实际上是多种变形的一个组合，包括 TimesRegular、TimesBold、TimesItalic、TimesOblique、TimesBoldItalic、TimesBoldOblique 等。Times 的每种变形都是一个具体的字体风格(Font Face)。

CSS 字体属性定义文本的字体系列、大小、加粗、风格(如斜体)和变形(如小型大写字母)。常用的字体属性及其说明如表 2-5 所示。

表 2-5　字体样式

属性名称	说　明
font	简写属性。作用是把所有针对字体的属性设置在一个声明中
font-family	设置字体系列
font-size	设置字体的尺寸

续表

属性名称	说　明
font-size-adjust	当首选字体不可用时，对替换字体进行智能缩放(CSS 2.1 已删除该属性)
font-stretch	对字体进行水平拉伸(CSS 2.1 已删除该属性)
font-style	设置字体风格
font-variant	以小型大写字体或者正常字体显示文本
font-weight	设置字体的粗细

font 这个简写属性用于一次设置元素字体的两个或更多方面。使用 icon 等关键字可以适当地设置元素的字体，使之与用户计算机环境中的某个方面一致。其所需要设置的字体如表 2-6 所示。

表 2-6　文本字体

字　体	说　明
caption	定义被标题控件(比如按钮、下拉列表等)使用的字体
icon	定义被图标标记使用的字体
menu	定义被菜单使用的字体
message-box	定义被对话框使用的字体
small-caption	caption 字体的小型版本
status-bar	定义被窗口状态栏使用的字体

在 font 属性中设置字体样式，如果没有使用表 2-6 中的字体关键词，至少要指定字体大小和字体系列。

font 属性能够将表 2-5 中的其他属性集合在一起，包括字体风格、大小和粗细等。其定义样式的顺序为 font-style、font-variant、font-weight、font-size、line-height、font-family。在介绍 font 属性定义字体样式之前，首先介绍这些样式及其取值。

1. font-style

font-style 属性设置字体使用斜体、倾斜或正常字体。斜体字体通常定义为字体系列中的一个单独的字体。该属性的取值及其说明如下所示。

- normal：默认值。浏览器显示一个标准的字体样式。
- italic：浏览器会显示一个斜体的字体样式。
- oblique：浏览器会显示一个倾斜的字体样式。
- inherit：规定应该从父元素继承字体样式。

如设置一个 div 使用字体为斜体的样式，代码如下：

```
<style type="text/css">
   div {
      font-style: italic;
   }
</style>
```

2. font-variant

font-variant 属性设置文本中大写字母的小型字体。通常大写字母与小写字母相比，除了形态不同，其字体大小也略有不同。

font-variant 属性设置小型大写字母的字体显示文本，这意味着所有的小写字母均会被转换为大写，但是所有使用小型大写字体的字母与其余文本相比，其字体尺寸更小。font-variant 属性可取值及其说明如下所示。

- normal：默认值。浏览器会显示一个标准的字体。
- small-caps：浏览器会显示小型大写字母的字体。
- inherit：规定从父元素继承 font-variant 属性的值。

3. font-weight

font-weight 属性设置文本的粗细。该属性用于设置显示元素的文本中所用的字体加粗。数值 400 相当于关键字 normal，700 等价于 bold。每个数值对应的字体加粗必须至少与下一最小数字一样细，且至少与下一最大数字一样粗。其属性值及说明如表 2-7 所示。

表 2-7　文本粗细

属 性 值	说　明
normal	默认值。定义标准的字符
bold	定义粗体字符
bolder	定义更粗的字符
lighter	定义更细的字符
100、200、...、900	定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold
inherit	规定应该从父元素继承字体的粗细

4. font-size

font-size 属性可设置字体的尺寸。该属性设置元素的字体大小，实际上它设置的是字体中字符框的高度；所显示的字符字形可能比这些框高或矮(通常会矮)。

font-size 属性值关键字对应的字体必须比一个最小关键字相应字体要高，并且要小于下一个最大关键字对应的字体。其属性值如表 2-8 所示。

表 2-8　字体尺寸

属 性 值	说　明
xx-small x-small small medium large x-large xx-large	把字体的尺寸设置为不同的尺寸，从 xx-small 到 xx-large 依次增大。默认为 medium

续表

属 性 值	说 明
smaller	把 font-size 设置为比父元素更小的尺寸
larger	把 font-size 设置为比父元素更大的尺寸
length	把 font-size 设置为一个固定的值
%	把 font-size 设置为基于父元素的一个百分比值
inherit	规定应该从父元素继承字体尺寸

5. line-height

line-height 属性设置行间的距离(行高)，该值不允许使用负值。line-height 属性会影响行框的布局。在应用到一个块级元素时，它定义了该元素中基线之间的最小距离而不是最大距离。line-height 属性的可取值如下所示。

- normal：默认值。设置合理的行间距。
- number：设置数值，此数值会与当前的字体尺寸相乘来设置行间距。
- length：设置固定的行间距。
- %：基于当前字体尺寸的百分比行间距。
- inherit：规定应该从父元素继承 line-height 属性的值。

6. font-family

font-family 规定元素的字体系列。font-family 可以把多个字体名称作为一个“回退”系统来保存。如果浏览器不支持第一个字体，则会尝试下一个。也就是说，font-family 属性的值是用于某个元素的字体族名称或/及类族名称的一个优先表。浏览器会使用它可识别的第一个值。在定义时使用逗号分割每个值，并始终提供一个类族名称作为最后的选择。

使用何种特定的字体系列(Geneva)完全取决于用户机器上该字体系列是否可用；这个属性没有指示任何字体下载。因此推荐使用一个通用字体系列名作为后路。常见的中文字体名称及其英文属性名如表 2-9 所示。

表 2-9 常用字体

字体名称	属 性 值	字体名称	属 性 值
华文楷体	STKaiti	标楷体	DFKai-SB
华文宋体	STSong	黑体	SimHei
华文中宋	STZhongsong	宋体	SimSun
华文仿宋	STFangsong	新宋体	NSimSun
方正舒体	FZShuTi	仿宋	FangSong
方正姚体	FZYaoti	楷体	KaiTi
华文彩云	STCaiyun	仿宋_GB2312	FangSong_GB2312
华文琥珀	STHupo	楷体_GB2312	KaiTi_GB2312
华文隶书	STLiti	隶书	LiShu

续表

字体名称	属 性 值	字体名称	属 性 值
华文行楷	STXingkai	幼圆	YouYuan
华文新魏	STXinwei	华文细黑	STXihei

7. font

font 属性能够将上述 6 种属性结合在一起，如定义一个 div 样式为斜体、字体大小为 16pt、行高为 20pt 的幼圆字体，代码如下：

```
<style type="text/css">
   .div1 {
       font: italic 16pt/20pt YouYuan;
    }
</style>
```

【例 2.6】

使用例 2.5 中图 2-5 给出的页面，修改其单元格的字体样式，分别使首行单元格显示宋体、最大字体、粗细为 500 大小；末行左侧单元格字体为斜体；末行中间单元格使用华文行楷字体；末行右侧单元格使用 italic、Times 或 serif 红色字体，字体大小为 12pt，行高为 14pt，步骤如下。

步骤 01 修改首行单元格显示宋体、最大字体、粗细为 500 大小，代码如下：

```
<style type="text/css">
    .td1 {
        letter-spacing: 1em;
        text-align: center;
        font-family: SimSun;
        font-size: xx-large;
        font-weight: 500;
    }
</style>
```

步骤 02 修改其末行左侧单元格的字体为斜体，代码如下：

```
.td2 {
    text-align: right;
    white-space: pre;
    font-style: italic;
}
```

步骤 03 修改末行中间单元格使用华文行楷字体，代码如下：

```
.td3 {
    text-align: center;
    white-space: pre-line;
    text-decoration: underline;
    font-family: STXingkai;
}
```

步骤 04 末行右侧单元格使用 italic、Times 或 serif 红色字体，字体大小为 12pt，行

高为 14pt。字体颜色使用 color 属性，该属性后面跟颜色的英文或颜色标记。使用 font 属性一次性定义，代码如下：

```
.td4 {
    text-align: left;
    color: red;
    font: italic 12pt/20pt Times, serif;
}
```

步骤 05 运行上述代码，其效果如图 2-6 所示。



图 2-6　使用字体样式

2.4.5 链接样式

链接也是一种文本，可设置其字体、颜色、背景等属性。但链接作为一种超文本，有着自己独特的样式，在 CSS 中根据链接的状态，可以将链接的样式分为 4 种，如下所示。

- a:link：普通的、未被访问的链接。
- a:visited：用户已访问的链接。
- a:hover：鼠标指针位于链接的上方。
- a:active：链接被点击的时刻。

上述 4 种状态的样式需要独立设置，但为链接的不同状态设置样式时，需要按照以下次序规则：

- a:hover 必须位于 a:link 和 a:visited 之后。
- a:active 必须位于 a:hover 之后。

【例 2.7】

创建页面并添加 4 个链接，分别使用不同样式定义链接的各种状态，步骤如下。

步骤 01 首先定义链接的 4 种样式，定义初始状态下链接字体为黑色；定义已经访问过的链接字体颜色为红色；定义鼠标移动到链接上方时，链接背景色为 #89D4F4；定义单击链接时，字体颜色为蓝色。代码如下：

```
<style type="text/css">
    a:link {
        color: black;
    }
    a:visited {
```

```
        color: red;
    }
    a:hover {
        background-color: #89D4F4;
    }
    a:active {
        color: blue;
    }
</style>
```

步骤 02 向页面的 body 元素下添加标题，定义 body 元素中的文本居中显示，并添加 4 个链接，如下所示：

```
<body style="text-align:center">
    <h2><a href="http://zz.meituan.com/">美团网</a></h2>
    <h2><a href="http://zz.nuomi.com/">糯米网</a></h2>
    <h2><a href="http://www.lashou.com/">拉手网</a></h2>
    <h2><a href="http://bj.jumei.com/">聚美优品</a></h2>
</body>
```

步骤 03 运行上述代码，并将鼠标放在“美团网”链接处，观察该链接有了背景色，如图 2-7 所示；单击“糯米网”链接后再返回该页面，其“糯米网”链接显示红色字体，如图 2-8 所示；将鼠标放在“拉手网”链接处并单击，其样式效果如图 2-9 所示。



图 2-7　链接访问前



图 2-8　链接访问后



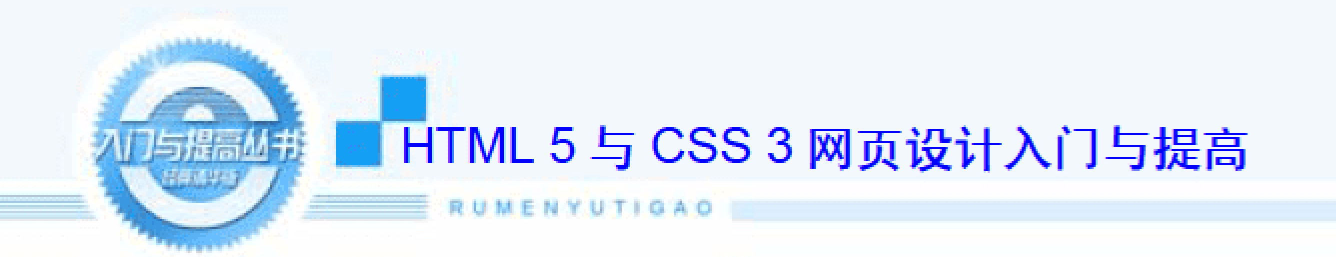
图 2-9　链接访问时

2.4.6 列表样式

列表是一种排列有序的文本，除了可以根据文本样式和字体样式来设置列表的样式，列表还有着独立的样式设置，如下所示。

- list-style：简写属性。把所有用于列表的属性设置于一个声明中。
- list-style-image：将图像设置为列表项标志。
- list-style-position：设置列表中列表项标志的位置。
- list-style-type：设置列表项标志的类型。

上述语句中，list-style 与其他属性之间的关系相当于 font 与 font-style、font-variant、font-weight、font-size、line-height、font-family 等属性之间的关系。



1. list-style-type

list-style-type 属性定义列表的标志类型。例如，在一个无序列表中，列表项的标志(marker)是出现在各列表项旁边的圆点。在有序列表中，标志可能是字母、数字或另外某种计数体系中的一个符号。

list-style-type 能够修改用于列表项的标志类型，其属性值如表 2-10 所示。

表 2-10　列表标志类型

属 性 值	说　明
none	无标记
disc	默认。标记是实心圆
circle	标记是空心圆
square	标记是实心方块
decimal	标记是数字
decimal-leading-zero	0 开头的数字标记(01、02、03 等)
lower-roman	小写罗马数字(i、ii、iii、iv、v 等)
upper-roman	大写罗马数字(I、II、III、IV、V 等)
lower-alpha	小写英文字母(a、b、c、d、e 等)
upper-alpha	大写英文字母(A、B、C、D、E 等)
lower-greek	小写希腊字母(alpha、beta、gamma 等)
lower-latin	小写拉丁字母
upper-latin	大写拉丁字母
hebrew	传统的希伯来编号方式
armenian	传统的亚美尼亚编号方式
georgian	传统的乔治亚编号方式(an、ban、gan 等)
cjk-ideographic	简单的表意数字
hiragana	标记是 a、i、u、e、o、ka、ki 等(日文片假名)
katakana	标记是 A、I、U、E、O、KA、KI 等(日文片假名)
hiragana-iroha	标记是 i、ro、ha、ni、ho、he、to 等(日文片假名)
katakana-iroha	标记是 I、RO、HA、NI、HO、HE、TO 等(日文片假名)

如定义无序列表的列表项标志，设置为方块，代码如下：

```
ul {
    list-style-type: square
}
```

2. list-style-image

常规的标志有时并不能满足用户的需求，人们更希望使用图片(如图标)来作为列表的标志，此时可以使用 CSS 中的 list-style-image 属性。

该属性可使用图片的地址作为属性值，在执行时，将图片作为列表标志来显示。如定义列表的标志图片为 xxx.gif，语法如下：

```
ul {
   list-style-image: url(xxx.gif)
}
```

3. list-style-position

list-style-position 属性用于声明列表标志相对于列表项内容的位置，其可取值如下。

- inside：列表项目标记放置在文本以内，且环绕文本根据标记对齐。
- outside：默认值。保持标记位于文本的左侧。列表项目标记放置在文本以外，且环绕文本不根据标记对齐。
- inherit：规定应该从父元素继承 list-style-position 属性的值。

4. list-style

为简单起见，可以将以上 3 个列表样式属性合并为一个方便的属性：list-style。

list-style 属性是一个简写属性，涵盖了所有其他列表样式属性。由于它应用到所有 display 为 list-item 的元素，所以在普通的 HTML 和 XHTML 中只能用于 li 元素，不过实际上它可以应用到任何元素，并由 list-item 元素继承。

list-style 的值可以按任何顺序列出，而且这些值都可以忽略。只要提供了一个值，其他的就会填入其默认值。语法如下所示：

```
li {
   list-style: url(example.gif) square inside
}
```

【例 2.8】

创建页面，添加一个两行三列的表格，其内容与例 2.5 相似。在第一行中间的单元格有文本“白居易”；在第二行的 3 个单元格中添加白居易的 3 首诗词，但这 3 首诗词以列表的样式来写，并为这 3 首诗词定义不同的列表样式，步骤如下。

步骤 01 定义 3 个列表样式，分别定义第 1 个为空心圆点标志；第 2 个使用图片标志；第 3 个设置列表项目标记放置在文本以内，且环绕文本根据标记对齐，代码如下：

```
<style type="text/css">
   .ul1 {
      list-style-type: circle;
   }
   .ul2 {
      list-style-image: url('Images/ul.jpg');
   }
   .ul3 {
      list-style-position: inside;
   }
</style>
```

步骤 02　修改例 2.5 中的代码，使其中的诗词部分以列表的形式呈现，代码如下：

```
<table>
   <tr>
      <td> </td>
      <td class="td1">白居易</td>
      <td> </td>
   </tr>
   <tr>
      <td class="td2">
         <ul class="ul1">
            <li>离离原上草，一岁一枯荣。</li>
            <li>野火烧不尽，春风吹又生。</li>
            <li>远芳侵古道，晴翠接荒城。</li>
            <li>又送王孙去，萋萋满别情。</li>
         </ul>
      </td>
      <td class="td3">
         <ul class="ul2">
            <li>帝城春欲暮，喧喧车马度。</li>
            <li>共道牡丹时，相随买花去。</li>
            <li>贵贱无常价，酬直看花数：</li>
            <li>灼灼百朵红，戋戋五束素。</li>
         </ul>
      </td>
      <td class="td4">
         <ul class="ul3">
            <li>江南好，风景旧曾谙。</li>
            <li>日出江花红胜火，春来江水绿如蓝，</li>
            <li>能不忆江南？
            </li>
         </ul>
      </td>
   </tr>
</table>
```

步骤 03　运行上述代码，执行效果如图 2-10 所示。



图 2-10　使用列表样式

2.4.7 表格和轮廓

表格与 div 一样，也是一种容器，可以添加标题、段落、文本和列表等。因此表格的样式可以根据文本样式、字体样式、背景样式等来设置。本节介绍表格中的特有样式，如表格的边距、单元格的算法、单元格合并等。

表格是可以有轮廓的，不只是表格，div 元素也有轮廓。本节除了介绍表格的样式，还将介绍轮廓的样式，包括轮廓的元素、宽度和样式等。

1. 表格的边框

表格的边框与轮廓在样式设置方面相似，但表格的边框和轮廓是不同的概念。边框是表格内部分离各个单元格的线，而轮廓是所有元素都可以有的、分布在元素周围的线。

表格的边框使用 border 属性，可设置边框的宽度、样式和颜色。由于 table、tr 以及 td 元素都有独立的边框，因此若同时为这 3 个元素设置边框，将出现单元格周围有多条边框的现象；如果需要把表格显示为单线条边框，需要使用 border-collapse 属性进行设置。

border-collapse 属性设置是否将表格边框折叠为单一边框，有以下几个属性值。

- separate：默认值。边框会被分开。不会忽略 border-spacing 和 empty-cells 属性。
- collapse：如果可能，边框会合并为一个单一的边框。会忽略 border-spacing 和 empty-cells 属性。

边框的设置有以下几个独立的属性，通过 border 属性可将下列独立的属性集合在一起。边框样式的属性如下所示。

- border-style：设置边框的样式。
- border-width：设置边框的宽度。
- border-color：设置边框的颜色。

border-style 属性用于设置元素所有边框的样式，或者单独地为各边设置边框样式。其属性值如表 2-11 所示。

表 2-11　边框的样式

属 性 值	说　明
none	定义无边框
hidden	与 none 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突
dotted	定义点状边框。在大多数浏览器中呈现为实线
dashed	定义虚线。在大多数浏览器中呈现为实线
solid	定义实线
double	定义双线。双线的宽度等于 border-width 的值
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值
inset	定义 3D inset 边框。其效果取决于 border-color 的值
outset	定义 3D outset 边框。其效果取决于 border-color 的值
inherit	规定从父元素继承边框样式

在表 2-11 列出的属性值中，最不可预测的边框样式是 double。它定义为两条线的宽度再加上这两条线之间的空间，等于 border-width 值。

注意：CSS 规范并没有说其中一条线是否比另一条粗或者两条线是否应该是一样粗，也没有指出线之间的空间是否应当比线粗。所有这些都由用户浏览器决定，开发人员对这个决定没有任何影响。

border-width 属性为元素的所有边框设置宽度，或者单独地为各边框设置宽度。只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

border-width 属性的可取值如下所示。

- thin：定义细的边框。
- medium：默认值。定义中等的边框。
- thick：定义粗的边框。
- length：允许您自定义边框的宽度。
- inherit：规定从父元素继承边框样式。

border-color 属性是一个简写属性，可设置四条边框的颜色。border-color 属性设置一个元素的所有边框中可见部分的颜色，或者为 4 个边分别设置不同的颜色。

border-color 属性的可取值与 color 属性一样，但 border-color 属性可取 1~4 个值，其取值的数量与定义的颜色样式如下所示。

- 定义 4 个颜色值：分别定义上边框、右边框、下边框和左边框的颜色。
- 定义 3 个颜色值：合并左右边框的颜色设置，分别定义上边框、左右边框和下边框的颜色。
- 定义 2 个颜色值：合并上下边框和左右边框的颜色设置，分别定义上下边框的颜色和左右边框的颜色。
- 定义 1 个颜色值：将所有边框的颜色统一设置为一种颜色。

颜色的取值可以是英文单词、十六进制颜色值、rgb 代码的颜色值或 transparent。默认值是 transparent，表示边框颜色为透明。

【例 2.9】

创建页面并添加两个表格，分别定义第 1 个表格的 4 个边框为虚线、粗线，颜色各不相同；定义第 2 个表格边框为 3D 垄状边框，上下边框颜色相同，左右边框颜色相同，步骤如下。

步骤 01 定义第 1 个表格的 4 个边框为虚线、粗线，颜色各不相同，定义第 2 个表格边框为 3D 垄状边框，上下边框颜色相同，左右边框颜色相同，其样式代码如下所示：

```
<style type="text/css">
   .table1 {
      border-style: dashed;
      border-width: thick;
      border-color: #89D4F4 #00ff90 #ffd800 #ff6a00;
```

```
    }
    .table2 {
        border-style: ridge;
        border-width: thick;
        border-color: #89D4F4 #00ff90;
    }
</style>
```

步骤 02 向页面添加两个表格，分别使用上述两种样式；向表格中添加诗词，省略部分代码，其表格代码如下所示：

```
<table>
    <tr>
        <td class="table1">
            <!--省略部分代码-->
        </td>
    </tr>
</table>
<table>
    <tr>
        <td class="table2">
            <!--省略部分代码-->
        </td>
    </tr>
</table>
```

步骤 03 运行上述代码，其效果如图 2-11 所示。

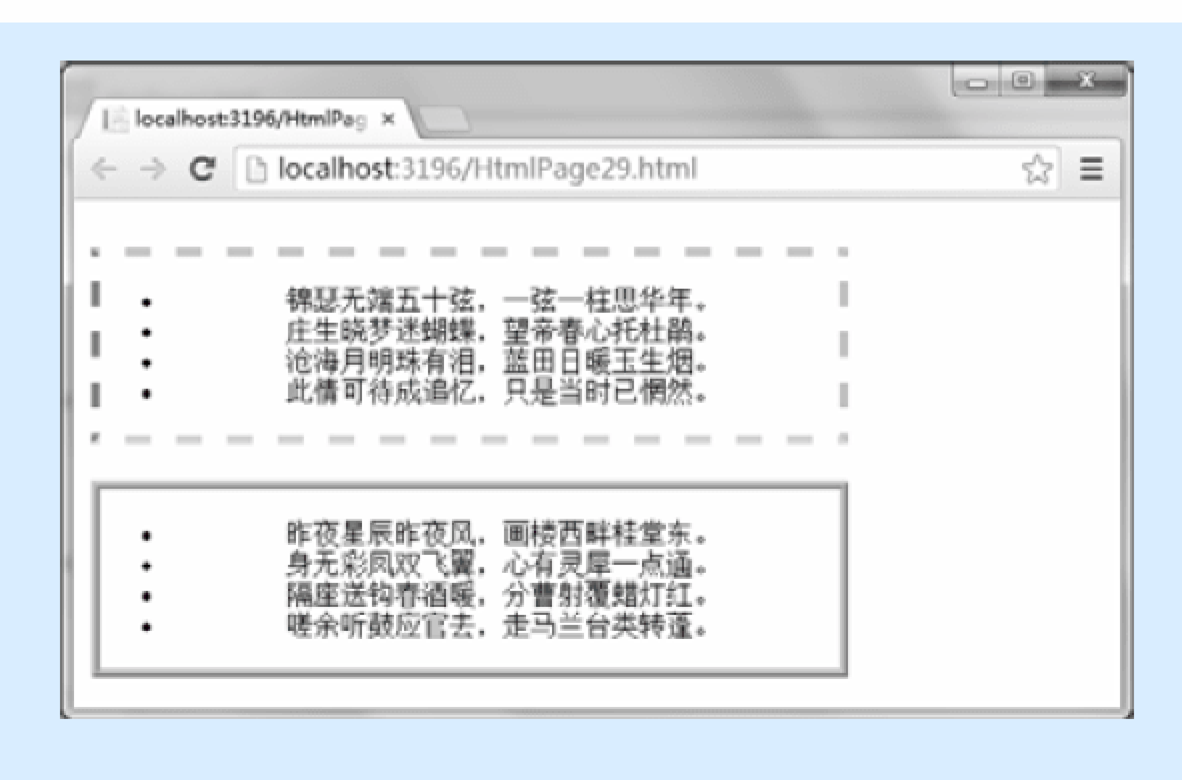


图 2-11　使用表格边框设置

2. 表格的其他样式

表格的其他样式包括表格的大小和边距、表格单元格的边框距离、标题位置、空白单元格的显示以及单元格算法等。

表格的大小由表格的宽度和高度决定，分别使用 width 属性和 height 属性来设置；表格的边距指的是表格的内边距，即表格内容与边框之间的距离，由 padding 属性来设置；单元格与单元格之间的距离使用 border-spacing 属性。width、height、padding、border-spacing 这几个属性都可以使用具体的数值或百分比来设置，这里不再详细介绍。

(1) 表格的 caption-side 属性设置表格标题的位置，有以下几种取值。

- top：默认值。把表格标题定位在表格之上。
- bottom：把表格标题定位在表格之下。
- inherit：规定从父元素继承 caption-side 属性的值。

(2) 表格的 empty-cells 属性设置是否显示表格中的空单元格。该属性定义了不包含任何内容的表单元格如何表示。如果显示，就会绘制出单元格的边框和背景。除非 border-collapse 设置为 separate，否则将忽略这个属性。

- hide：不在空单元格周围绘制边框。
- show：在空单元格周围绘制边框。默认值。
- inherit：规定应该从父元素继承 empty-cells 属性的值。

(3) table-layout 设置单元、行和列的算法规则，实质是设置单元格的大小怎样决定。单元格的大小可以由浏览器根据内容自动设置，也可以由开发人员设置。单元格的设置有两种，一种是固定设置，由开发人员固定单元格的大小；另一种是自动设置。说明如下：

- 固定设置与自动设置相比，允许浏览器更快地对表格进行设置。
- 在固定设置中，水平大小仅取决于表格宽度、列宽度、表格边框宽度、单元格间距，而与单元格的内容无关。
- 通过使用固定设置，浏览器在接收到第一行后就可以显示表格。
- 在自动设置中，列的宽度是由列单元格中没有折行的最宽的内容设定的。此算法有时会较慢，这是由于它在确定最终的设置之前需要访问表格中所有的内容。
- 固定设置算法比较快，但是不太灵活，而自动设置比较慢，不过更能反映传统的 HTML 表格。

3. 轮廓

轮廓的表现形式与边框类似，因此其属性也相似，都有着定义线条宽度、线条样式和颜色的属性。

outline-style 属性设置元素的轮廓样式，其属性值与边框的样式相似，如表 2-12 所示。

表 2-12 轮廓样式

属 性 值	说 明
none	默认值。定义无轮廓
dotted	定义点状的轮廓
dashed	定义虚线轮廓
solid	定义实线轮廓
double	定义双线轮廓。双线的宽度等同于 outline-width 的值
groove	定义 3D 凹槽轮廓。此效果取决于 outline-color 值
ridge	定义 3D 凸槽轮廓。此效果取决于 outline-color 值
inset	定义 3D 凹边轮廓。此效果取决于 outline-color 值
outset	定义 3D 凸边轮廓。此效果取决于 outline-color 值
inherit	规定从父元素继承轮廓样式的设置

除了样式以外，轮廓使用 outline-color 属性设置颜色，其属性值可参考边框颜色的属性值；使用 outline-width 属性设置宽度，其属性值可参考边框宽度的属性值。

2.4.8 其他样式

在 HTML 元素中，很多元素都有着边框和内容之间的距离(内边距)，以及围绕元素边框的空白区域(外边距)。除此之外，元素还具有定位属性和浮动属性。

1. 内边距

内边距使用 padding 属性定义，其属性值可以是 auto、长度值或百分比值，但不允许使用负值。auto 表示使用浏览器默认的样式。

内边距分为上边距、下边距、左边距和右边距，可以为 padding 属性定义 4 个属性值，分别根据上、右、下、左的顺序为元素设置内边距；也可以使用 padding-top、padding-right、padding-bottom、padding-left 这 4 个属性分别对上边距、右边距、下边距、左边距进行设置。

这些边距的属性值可以是百分比，百分数值是相对于其父元素的 width 计算的，所以如果父元素的 width 改变，它们也会改变。

上下内边距与左右内边距一致，即上下内边距的百分数会相对于父元素宽度设置，而不是相对于高度。

2. 外边距

设置外边距会在元素外创建额外的“空白”。设置外边距最简单的方法就是使用 margin 属性，该属性接受任何长度单位、百分数值、auto，甚至是负值。

margin 属性与 padding 属性一样，可以同时定义 4 个方向上的外边距，根据上、右、下、左的顺序为元素设置外边距；也可以分别使用 margin-bottom(元素的下外边距)、margin-left(元素的左外边距)、margin-right(元素的右外边距)、margin-top(元素的上外边距)来设置外边距。

padding 属性和 margin 属性都可以有缺省的情况，即定义少于 4 个边距值，来设置 4 个边距。其使用规则如下所示：

- 如果缺少左外边距的值，则使用右外边距的值。
- 如果缺少下外边距的值，则使用上外边距的值。
- 如果缺少右外边距的值，则使用上外边距的值。

换句话说，如果为外边距指定了 3 个值，则第 4 个值(即左外边距)会从第 2 个值(右外边距)复制得到。如果给定了两个值，第 4 个值会从第 2 个值复制得到，第 3 个值(下外边距)会从第 1 个值(上外边距)复制得到。最后一种情况，如果只给定一个值，那么其他 3 个外边距都由这个值(上外边距)复制得到。

Netscape 和 IE 对 body 标记定义的默认边距(margin)值是 8px。而 Opera 不是这样。相反地，Opera 将内部填充(padding)的默认值定义为 8px，因此如果希望对整个网站的边缘部分进行调整，并将之正确显示于 Opera 中，那么必须对 body 的 padding 进行自定义。

3. 外边距的合并

与表格中边框的合并一样，外边距是可以合并的。但这个合并是由浏览器自主控制，并不是人为设置的。

通常，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

外边距合并(叠加)是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。外边距的合并有以下几种情况：

- 当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。
- 当一个元素包含在另一个元素中时(假设没有内边距或边框把外边距分隔开)，它们的上和/或下外边距也会发生合并。
- 假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并。如果这个外边距遇到另一个元素的外边距，它还会发生合并。这就是一系列的段落元素占用空间非常小的原因，因为它们的所有外边距都合并到一起，形成了一个小的外边距。

以由几个段落组成的典型文本页面为例，第一个段落上面的空间等于段落的上外边距。如果没有外边距合并，后续所有段落之间的外边距都将是相邻上外边距和下外边距的和。这意味着段落之间的空间是页面顶部的两倍。如果发生外边距合并，段落之间的上外边距和下外边距就合并在一起，这样各处的距离就一致了。

只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

4. 元素的定位与浮动

CSS 为定位和浮动提供了一些属性，利用这些属性，可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务。

定位的基本思想很简单，它允许定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。

另一方面，CSS 1 中首次提出了浮动，它以 Netscape 在 Web 发展初期增加的一个功能为基础。浮动不完全是定位，不过，它当然也不是正常流布局。我们会在后面的章节中明确浮动的含义。

元素的定位使用position属性，通过使用position属性，可以选择 4 种不同类型的定位，这会影响元素框生成的方式。position属性值及其说明如表 2-13 所示。

表 2-13　position属性(元素框生成的方式)

属 性 值	说　明
static	元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中
relative	元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留

续表

属性值	说明
absolute	元素框从文档流完全删除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框
fixed	元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身

提示：相对定位实际上被看作普通流定位模型的一部分，因为元素的位置是相对于它在普通流中的位置的。

在 position 属性定义了元素框生成方式之后，即可对元素的位置进行设置，包括顶部偏移、左侧偏移、右侧偏移等，如表 2-14 所示。

表 2-14　position属性(元素的位置)

属性值	说明
top	定义了一个定位元素的上外边距边界与其包含块上边界之间的偏移
right	定义了定位元素右外边距边界与其包含块右边界之间的偏移
bottom	定义了定位元素下外边距边界与其包含块下边界之间的偏移
left	定义了定位元素左外边距边界与其包含块左边界之间的偏移
overflow	设置当元素的内容溢出其区域时发生的事情
clip	设置元素的形状。元素被剪入这个形状之中，然后显示出来
vertical-align	设置元素的垂直对齐方式
z-index	设置元素的堆叠顺序

表 2-14 中，上面 4 个偏移属性可直接设置长度、百分比等数值，overflow 表示当内容溢出时对内容的剪辑和显示形式，可取值如下所示。

- visible：默认值。内容不会被修剪，会呈现在元素框之外。
- hidden：内容会被修剪，并且其余内容是不可见的。
- scroll：内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。
- auto：如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。
- inherit：规定应该从父元素继承 overflow 属性的值。

clip 属性用于定义一个剪裁矩形。对于一个绝对定义元素，在这个矩形内的内容才可见。出了这个剪裁区域的内容会根据 overflow 的值来处理。剪裁区域可能比元素的内容区大，也可能比内容区小。clip 属性主要对图片与区域大小不符的情况进行定义，可取值为 top、right、bottom、left。

上述属性定义了元素的定位，CSS 提供了浮动属性来设置元素的浮动效果，使用 float 属性。

float 属性定义元素在哪个方向浮动。以往这个属性总应用于图像，使文本围绕在图像周围，不过在 CSS 中，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。

如果浮动非替换元素，则要指定一个明确的宽度；否则，它们会尽可能地窄。

假如在一行之上只有极少的空间可供浮动元素，那么这个元素会跳至下一行，这个过程会持续到某一行拥有足够的空间为止。

- left：元素向左浮动。
- right：元素向右浮动。
- none：默认值。元素不浮动，并会显示在其在文本中出现的位置。
- inherit：规定从父元素继承 float 属性的值。

【例 2.10】

创建页面并添加 4 个 div 块，为其定义相同的长度和宽度，相邻的两个 div 块使用不同的背景色。分别查看这 4 个 div 块的默认定位和添加浮动属性后的位置，步骤如下。

步骤 01 定义 3 个选择器样式，其中 1 个是 div 样式，定义所有的 div 使用相同的长度和宽带；另外定义两个选择器，为 div 设置不同的背景色。相邻的两个 div 使用不同的背景色，那么 4 个 div 使用两种背景色即可。在定义了背景色之后，为 div 设置浮动属性并对该属性进行注释，代码如下：

```
<style type="text/css">
    div {
        width: 100px;
        height: 25px;
    }
    .div1 {
        background-color: #0ff;
        /*float: left;*/
    }
    .div2 {
        background-color: #0094ff;
        /*float: left;*/
    }
</style>
```

步骤 02 向页面添加 4 个 div 并使相邻的两个用不同的选择器样式，代码如下：

```
<div class="div1"> </div>
<div class="div2"> </div>
<div class="div1"> </div>
<div class="div2"> </div>
```

上述代码中，div 除了背景色、出现的顺序不同外，其他设置完全相同。其显示效果如图 2-12 所示。

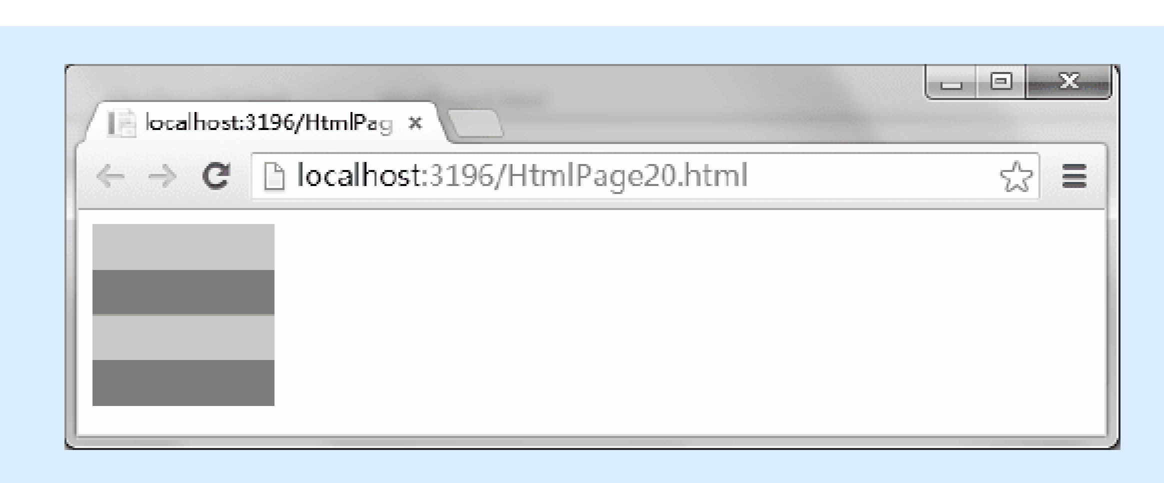


图 2-12 div的默认位置

步骤03 将步骤 01 中的注释改为非注释状态，运行上述代码，显示效果如图 2-13 所示。由于 div 宽度较小，完全可以并排显示在同一行，因此使用了靠左浮动属性的 div 顺序排列在了页面的同一个高度。

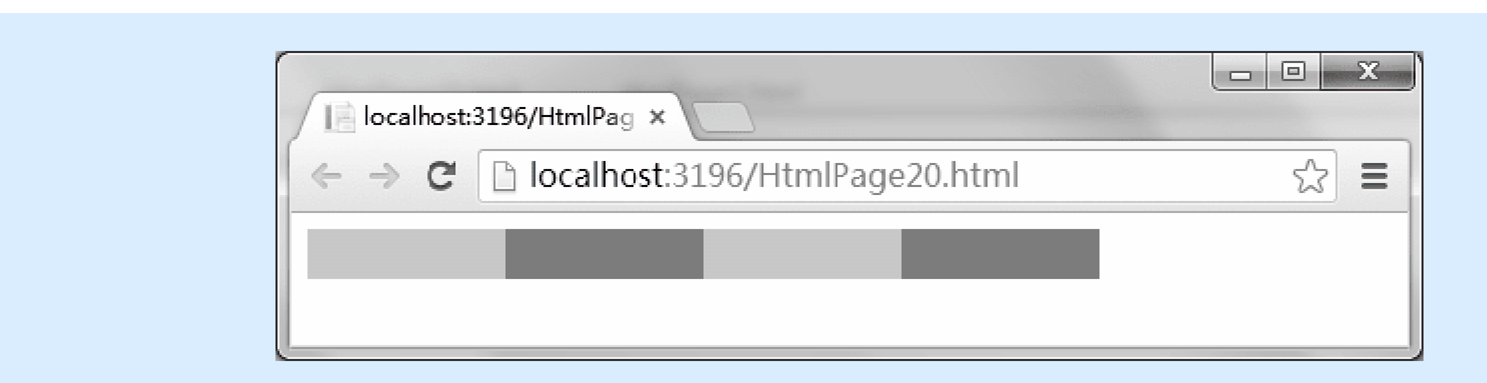


图 2-13 div的浮动效果

2.5 实战——诗词鉴赏页面设计

本章前面部分详细介绍了 CSS 的用法及其常用样式，这里结合本章的知识，设计诗词鉴赏页面。使用例 2.8 中的页面构成，向其中添加两行表格，分别为“李商隐”文本和 3 首李商隐的诗词，并分别为这个表格以及表中单元格内的列表设置样式，步骤如下。

步骤01 定义表格的样式。首先定义所有的表格宽度为 30%，字体居中显示，代码如下：

```
<style type="text/css">
    td {
        width: 30%;
        text-align: center;
    }
</style>
```

步骤02 定义单个单元格的样式。定义“白居易”和“李商隐”这两行使用相同的样式：字间距 1em；字体为 SimSun；字体大小为 xx-large；字体粗细为 500，代码如下：

```
.td1 {
    letter-spacing: 1em;
    font-family: SimSun;
    font-size: xx-large;
    font-weight: 500;
}
```

步骤03 定义白居易诗词行的第 1 个单元格字体为红色，代码如下：

```
.td2 {
    color: red;
}
```

步骤04 定义白居易诗词行的第 2 个单元格字体为 STXingkai，代码如下：

```
.td3 {
    font-family: STXingkai;
}
```

步骤 05 为白居易诗词行的第 3 个单元格字体添加下划线，代码如下：

```
.td4 {
    text-decoration: underline;
}
```

步骤 06 定义李商隐诗词行的第 1 个单元格字体是斜体，代码如下：

```
.td5 {
    font-style: italic;
}
```

步骤 07 设置李商隐诗词行的第 3 个单元格字体是蓝色，代码如下：

```
.td6 {
    color: blue;
}
```

步骤 08 设置两个列表的图片标志，分别用于白居易和李商隐的诗词，代码如下：

```
.ul1 {
    list-style-image: url('Images/star.jpg');
}
.ul2 {
    list-style-image: url('Images/starblue.jpg');
}
```

步骤 09 向页面中添加表格，分别在表格的每一行(即 tr 元素)中添加内联样式，定义“白居易”和“李商隐”这两行使用“#89D4F4”背景色；而两人的诗词行使用“#CCFFFF”背景色，并分别为每一个单元格添加 class 属性加载 CSS 样式。这里省略页面中的诗词文本，代码如下：

```
<table>
    <tr style="background-color: #89D4F4">
        <td> </td>
        <td class="td1">白居易</td>
        <td> </td>
    </tr>
    <tr style="background-color: #CCFFFF">
        <td class="td2">
            <ul class="ul1">
                <!--省略诗词文本-->
            </ul>
        </td>
        <td class="td3">
            <ul class="ul1">
                <!--省略诗词文本-->
            </ul>
        </td>
        <td class="td4">
            <ul class="ul1">
                <!--省略诗词文本-->
            </ul>
        </td>
    </tr>
```

```
    <tr style="background-color: #89D4F4">
        <td> </td>
        <td class="td1">李商隐</td>
        <td> </td>
    </tr>
    <tr style="background-color: #CCFFFF">
        <td class="td5">
            <ul class="ul2">
                <!--省略诗词文本-->
            </ul>
        </td>
        <td>
            <ul class="ul2">
                <!--省略诗词文本-->
            </ul>
        </td>
        <td class="td6">
            <ul class="ul2">
                <!--省略诗词文本-->
            </ul>
        </td>
    </tr>
</table>
```

步骤 10　运行上述代码，效果如图 2-14 所示。



图 2-14　诗词鉴赏

2.6　本 章 习 题

1. 填空题

(1)　CSS 样式有着 3 种形式：外部样式表、内部样式表和________。

(2)　选择器分组时使用________将选择器隔开。

(3)　设置背景不重复的语句是________。

(4) 设置字体为黑体，使用________属性。
(5) 设置字体为黑体，使用________属性值。
(6) 设置已访问过的链接使用________属性。

2. 选择题

(1) 下列属性值中，background-position 属性不能取的值是________。
A. top left B. top right C. bottom left D. top bottom
(2) 下列不属于文本对齐方式 text-align 取值的是________。
A. top B. center C. inherit D. justify
(3) 下列字体样式中，在 CSS 2.1 中已删除的是________。
A. font-size B. font-stretch C. font-style D. font-variant
(4) 下列关于链接的属性错误的是________。
A. a:link 表示未被访问的链接
B. a:visited 表示用户已访问的链接
C. a:hover 表示鼠标指针位于链接的上方
D. a:active 表示普通的链接
(5) 列表的默认标志类型是________。
A. disc B. circle C. square D. decimal
(6) 下列关于选择器分组的样式语法正确的是________。
A. h1,2,3{} B. h1;2;3{} C. h1,h2,h3{} D. h1;h2;h3{}

3. 上机练习

结合本章知识，设计新闻页面。要求新闻页面以表格的形式显示当前新闻分类，将页面分为若干单元格，每个单元格有新闻标题构成的链接，每个单元格是一个类型的新闻。

如娱乐新闻的新闻链接以列表的形式放在一个单元格；体育新闻的新闻链接以列表的形式放在另一个单元格。

新闻页面的设计要符合下面的要求：

- 不同类型的新闻使用不同列表标志图片。
- 表格的奇数行和偶数行使用不同的背景色。
- 要求没有访问过的链接使用黑色字体；访问过的使用红色字体；鼠标放在链接上面时，链接显示粉色背景。
- 页面中的列表统一靠左显示。
- 页面中的字体统一使用仿宋字体。





第3章

JavaScript脚本语言

网页技术中的 HTML 和 CSS 共同实现了页面设计，但若要使页面呈现动态效果，离不开脚本语言。

脚本是批处理文件的延伸，是一种以纯文本方式保存的程序。计算机脚本程序通常是确定的一系列控制计算机进行运算操作动作的组合，在其中可以实现一定的逻辑分支等。

JavaScript 是世界上最流行的脚本语言。本章详细介绍 JavaScript 的基础知识、优点及其应用。

本章学习目标：

- 了解 JavaScript 脚本的特点
- 理解 JavaScript 变量的使用
- 掌握 JavaScript 常用数据类型
- 掌握 JavaScript 运算符的使用
- 掌握 JavaScript 语句的使用
- 了解 JavaScript 对象
- 理解 JavaScript 函数
- 掌握 JavaScript 对象和函数的使用

3.1 JavaScript 脚本概述

JavaScript 是一种基于对象和事件驱动并具有相对安全性的客户端脚本语言。同时也是一种广泛应用于客户端 Web 开发的脚本语言，常用来给 HTML 网页添加动态功能，比如响应用户的各种操作。

JavaScript 是客户端脚本语言，不同于服务器端脚本语言的是，JavaScript 是在用户的浏览器上运行的，不需要服务器的支持即可以独立运行。所以在早期，程序员比较青睐于 JavaScript 的原因，是可以减少对服务器的负担。

1. JavaScript的特点

由于 JavaScript 是运行在客户端的，因此其安全性是程序员担忧的问题。尽管如此，JavaScript 仍然以其跨平台、容易上手等优势大行其道。

JavaScript 是世界上最流行的编程语言，其优点如下：

- JavaScript 是属于 Web 的语言，它适用于服务器、PC、笔记本电脑、平板电脑和智能手机等设备。
- JavaScript 是一种轻量级的编程语言。
- JavaScript 是可插入 HTML 页面的编程代码。
- JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。
- JavaScript 容易学习。几乎每个人都能将小的 JavaScript 片段添加到网页中。
- 客户端脚本在客户端执行，减轻了服务器的负担。

JavaScript 是一种脚本语言，其源代码在发往客户端运行之前不需经过编译，而是将文本格式的字符代码发送给浏览器，由浏览器解释运行。这样的语言称为解释语言。

解释语言也有着它们的弱点，表现如下：

- 安全性较差。
- 如果一条 JavaScript 语句运行不了，那么其后续的语句也无法运行。
- 每次重新加载都会重新解释，速度较慢。

2. JavaScript的构成

一个完整的 JavaScript 实现由以下 3 个不同部分组成：核心(ECMAScript)、文档对象模型(Document Object Model，DOM)、浏览器对象模型(Browser Object Model，BOM)。

JavaScript 是一种程序设计语言，有着自己的变量、数据类型、语句、函数和对象。JavaScript 程序是由若干语句组成的，语句是编写程序的指令。

JavaScript 提供了完整的基本编程语句，它们是赋值语句、switch 选择语句、while 循环语句、for 循环语句、foreach 循环语句、do-while 循环语句、break 循环中止语句、continue 循环中断语句、with 语句、try-catch 语句、if 语句(if-else，if-else-if)。

JavaScript 虽然是弱类型的程序设计语言，但其内置的对象能够处理不同类型的数据，其常见的数据类型有对象、数组、数、布尔值、空值；而 JavaScript 可使用的数据处理有

字符串处理、日期处理、数组处理、逻辑处理、算术处理等。

程序设计语言中通常都有运算符的使用，JavaScript 中的运算符与其他程序设计语言一样，有算术运算符、比较运算符、字符串连接符、逻辑运算符和三目运算符。

3.2 JavaScript 的基本语法

了解了 JavaScript 的基础知识以后，本节开始介绍 JavaScript 的基本语法，包括一个简单的 JavaScript 示例、JavaScript 中的输出和注释等。

3.2.1 简单的JavaScript例子

JavaScript 语句运行在客户端，需要嵌入在 HTML 中借助浏览器来执行。JavaScript 可以以语句的形式直接嵌入 HTML 内部；也可以在 HTML 中引用外部的 JavaScript 文件。

HTML 中的脚本代码必须放在<script>与</script>标记之间。脚本可被放置在 HTML 页面的<body>和<head>部分中。

如需在 HTML 页面中插入 JavaScript，需要使用<script>标记，其开始标记<script>和结束标记</script>会告诉 JavaScript 在何处开始和结束；而 JavaScript 代码放在<script>和</script>标记之间。浏览器会解释并执行位于<script>和</script>之间的 JavaScript。

由于 HTML 中的脚本语言不止一种，因此在 HTML 之前的版本中的<script>标记中有着 type="text/JavaScript"属性，该属性表示该<script>标记是 JavaScript 脚本语言。而当前的浏览器以及 HTML 5 中默认使用 JavaScript 脚本语言，因此该属性可以省略。

【例 3.1】

JavaScript 脚本语言有着可以直接使用的函数，如 document.write()方法可直接向页面中输出文本。本示例将 JavaScript 脚本放在<body>部分，向页面输出一个标题和一个段落，代码如下：

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
   <script>
     document.write("<h1>HELLO JavaScript</h1>");
     document.write("<p>This is a JavaScript</p>");
   </script>
</body>
</html>
```

上述代码的执行效果如图 3-1 所示。

在 HTML 文档中可以放入不限数量的脚本。脚本可位于 HTML 的<body>或<head>部分中，或者同时存在于两个部分中。通常的做法是把函数放入<head>部分中，或者放在页面底部。这样就可以把它们安置到同一处位置，而不会干扰页面的内容。



图 3-1　向<body>中写文本

脚本也可以保存到外部文件中，并在 HTML 文档中引用。外部文件通常包含被多个网页使用的代码，即多个页面可共同使用的功能。

外部的 JavaScript 脚本的实际运行效果与在 HTML 内部<script>标记中编写脚本完全一致。而且，<head>或<body>中都可以引用外部脚本文件。

外部 JavaScript 文件的文件扩展名是.js，如需使用外部文件，需要在<script>标记的 src 属性中设置该.js 文件，代码如下：

```
<script src="myScript.js"></script>
```

【例 3.2】

同时使用内部 JavaScript 脚本和外部 JavaScript 文件，在例 3.1 的基础上引用外部文件，步骤如下。

步骤 01　创建 JavaScript 文件 JavaScript1.js 并写入输出语句，代码如下：

```
document.write("<h2>read JavaScript1.js</h2>");
document.write("<p>write JavaScript1.js</p>");
```

步骤 02　修改例 3.1 中的页面代码如下：

```
<!DOCTYPE html>

<html>
<head>
</head>
<body>
    <script>
        document.write("<h1>HELLO JavaScript</h1>");
        document.write("<p>This is a JavaScript</p>");
    </script>
    <script src="JavaScript1.js"></script>
</body>
</html>
```

步骤 03　在浏览器中查看该页面，其效果如图 3-2 所示。



注意：外部脚本文件中不能包含<script>标记。



图 3-2　引用外部脚本的执行效果

3.2.2　JavaScript语句

程序都是由语句构成的，JavaScript 也是一样。除了本章 3.1 节介绍的基本语句以外，JavaScript 中有着可直接调用的对象和函数，用来实现特定的功能。

例如，例 3.1 中的 document.write()方法是一个输出方法，该语句是一条输出语句。本节介绍 JavaScript 中的语句规则及其常用语句。

1. 语句规则

JavaScript 语句有着自己的执行规则，有着分号、大小写字母、空格和换行等这些符号的使用规则，具体如下所示。

(1)　分号

分号用于分隔 JavaScript 语句，通常可以在每条可执行的语句结尾处添加分号。在 JavaScript 中，用分号来结束语句是可选的，分号在 JavaScript 中的作用是在一行中编写多条语句。因此分号和换行都可以作为 JavaScript 语句的结束。

(2)　JavaScript 对大小写敏感

JavaScript 对大小写是敏感的，因此在编写 JavaScript 语句时，应留意是否关闭了大小写切换键。例如，函数 getElementById()与 getElementbyID()是不同的；变量 myVariable 与 MyVariable 也是不同的。

(3)　空格

JavaScript 会忽略多余的空格。通常可以向脚本中添加空格，来提高其可读性。

(4)　单引号和双引号

在 JavaScript 中，单引号和双引号没有特殊的区别，都可以用来创建字符串，但是一般情况下 JavaScript 使用单引号。

在 JavaScript 中，单引号里面可以有双引号，双引号里面也可以有单引号。

特殊情况下 JavaScript 需要使用转义符号“\”，例如用(\")表示("),用(\')表示(')，而在 HTML 中则是用&quot。

(5)　语句块

JavaScript 中也有着语句块的概念，语句块是多条语句的结合，JavaScript 中的语句块使用大括号来定义开始和结束。

2. 输出语句

JavaScript 中的输出函数不止 document.write()这一个，document.write()函数是向页面中输出文本或 HTML 标记，使用 JavaScript 还可向页面中输出对话框。

(1) prompt()函数

此函数首先在浏览器窗口中弹出一个对话框，让用户自行输入信息。一旦输入完成后，就返回用户所输入信息的值。通过使用 JavaScript 中所提供的窗口对象函数 prompt()，就能完成信息的输入。该函数提供了最简便的信息输入方式，其基本使用格式如下：

```
test = prompt("请输入数据: ", "this is a JavaScript")
```

实际上 prompt()是 window 对象的一个函数。因为默认情况下所用的对象就是 window 对象，所以 window 对象可以省略不写。

(2) document.write()函数和 document.writeln()函数

document 是 JavaScript 中的一个对象，在其中封装了许多有用的函数，其中 write()和 writeln()就是用于将文本信息直接输出到浏览器窗口中的函数。二者的唯一区别就是 writeln()函数自动在文本之后加入回车符。

(3) window.alert()函数

JavaScript 为了方便信息输出，提供了具有独立对话框信息输出的 alert()函数。alert()函数是 window 对象的一个函数，它的主要用途是在输出时产生有关警告提示信息或提示用户，一旦用户单击“确定”按钮后，将会继续执行其他脚本程序。

【例 3.3】

使用 document.writeln()函数和 alert()函数，分别向页面输出文本和对话框，代码如下：

```
<!DOCTYPE html>
<html>
<head></head>
<body>
    <script>
        document.writeln("<h1>HELLO JavaScript</h1>");
        alert("This is a JavaScript");
    </script>
</body>
</html>
```

运行该页面，效果如图 3-3 所示。

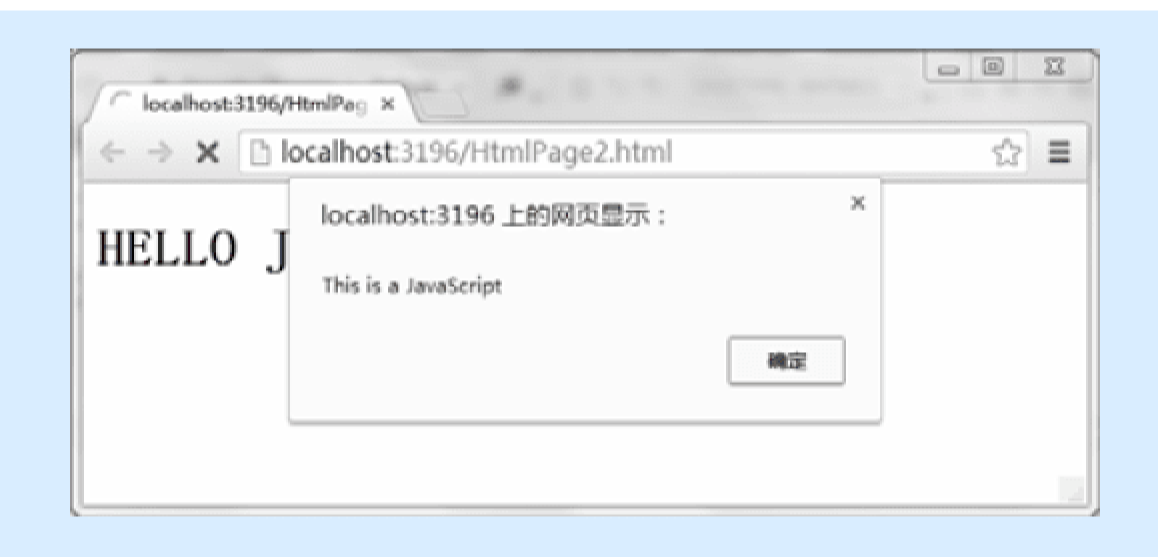


图 3-3　页面输出

注意：document.write()仅仅用于向文档写输出内容，如果在文档已完成加载后执行 document.write()，整个 HTML 页面将被覆盖。

3. 注释

大多数程序设计语言都有着注释语句，注释可用于提高代码的可读性，注释语句是不会被执行的。

JavaScript 中的注释有两种，一种是单行注释，使用双斜杠“//”；另一种是多行注释，将需要注释的文本放在“/*”和“*/”之间。

(1) 单行注释

单行注释有着两种类型，一种是整行注释，将双斜杠放在语句的开头；另一种是在语句后面添加的注释，只注释后面的解释语句。

对整行的注释和对语句的注释如下所示：

```
//这是整行注释
document.write("JavaScript")  //输出 JavaScript 文本，这是对语句的注释
```

上述代码中，第 2 行语句中“//”符号前面的语句是可以被解析的，而“//”后面的内容将被忽略。

注意：单行注释是不能够换行的，若需要换行的话，要在每一行注释的开头都用双斜杠标注。

(2) 多行注释

多行注释用于解释语句较多的情况。多行语句注释并不是只能够注释多行，也可以在单行进行注释。多行注释使用开始符号“/*”和结束符号“*/”，注释文本放在两个符号之间。这两个符号可放在同一行，也可放在不同的行，放在同一行相当于单行的注释，如下所示：

```
/* 单行注释 */
/* 多行注释第 1 行
多行注释第 2 行 */
```

3.3 JavaScript 变量

程序开发通常都有着变量的使用，变量取代程序中的数据值，参与程序的执行。在程序开发的过程中，变量的使用使程序有了重用性和可移植性，几乎每一个函数都需要有变量的参与。本节介绍 JavaScript 中的变量。

3.3.1 变量

程序中的变量可以理解为数学中的变量，在数学函数中，x 是一个没有确定值的数，

如椭圆函数、抛物线函数、双曲线函数等，图形根据 x 的值而变化，程序中的变量也是如此。变量由两部分构成，一个是变量的名称，另一个是变量的值。如 x=56，则变量名称为 x，值为 56。

变量存储可以变化的值，变量可以被声明，可以被初始化，可以赋值。变量的命名由用户决定，通常命名为有实际意义的词汇。

JavaScript 中的变量命名有如下几个规则：

- 变量必须以字母、_和$符号开头。
- 变量名称对大小写敏感(例如 y 和 Y 是不同的变量)。
- 变量名称不能是保留字。

大多程序开发语言在声明变量时需要为变量指定一个数据类型，JavaScript 变量有很多种类型，但在声明时并不指定。JavaScript 使用 var 关键词来声明变量，如下所示：

```
var name;
```

变量声明之后，该变量是空的(它没有值)。如需向变量赋值，可以使用等号，例如：

```
name = "Lucy";
```

变量在声明时也可以直接赋值，例如：

```
var name = "Lucy";
```

当为变量分配文本值时，应该用双引号或单引号包围这个值，否则变量的值将可能被解释为数值。向变量赋的值是数值时，不要使用引号。如果用引号包围数值，该值会被作为文本来处理。

JavaScript 可以同时声明多个变量，各个变量之间使用逗号分隔，如下所示：

```
var name="Gates", age=56, job="CEO";
```

声明也可跨越多行：

```
var name="Gates",
age=56,
job="CEO";
```

在计算机程序中，经常会声明无值的变量。没有赋值的变量，其值实际上是 undefined。如果声明一个变量但不赋值，那么输出该变量时，其值显示为 undefined。

3.3.2 数据类型

JavaScript 中有 5 种简单的数据类型：Undefined、Null、Boolean、Number 和 String。还有一种复杂数据类型 Object，Object 本质上是由一组无序的值对组成的。

1. Undefined类型

Undefined 类型只有一个值，即特殊的 undefined。在使用 var 声明变量但未对其加以初始化时，这个变量的值就是 undefined。

2. Null类型

Null 类型是第二个只有一个值的数据类型，这个特殊的值是 null。从逻辑角度来看，null 值表示一个空对象指针，这正是使用 typeof 操作符检测 null 时会返回 object 的原因。

如果定义的变量准备在将来用于保存对象，那么最好将该变量初始化为 null，而不是其他值。这样一来，只要直接检测 null 值，就可以知道相应的变量是否已经保存了一个对象的引用了。

实际上，undefined 值是派生自 null 值的，因此 ECMA-262 规定对它们的相等性测试要返回 true。

尽管 null 和 undefined 有这样的关系，但它们的用途完全不同。无论在什么情况下都没有必要把一个变量的值显式地设置为 undefined，可是同样的规则对 null 却不适用。换句话说，只要是保存对象的变量还没有真正保存对象，就应该明确地让该变量保存 null 值。这样做不仅可以体现 null 作为空对象指针的惯例，而且也有助于进一步区分 null 和 undefined。

3. Boolean类型

该类型只有两个字面值：true 和 false。这两个值与数字值不是一回事，因此 true 不一定等于 1，而 false 也不一定等于 0。

虽然 Boolean 类型的字面值只有两个，但 JavaScript 中所有类型的值都有与这两个 Boolean 值等价的值。要将一个值转换为其对应的 Boolean 值，可以调用类型转换函数 Boolean()。

可以对任何数据类型的值调用 Boolean()函数，而且总会返回一个 Boolean 值。至于返回的这个值是 true 还是 false，取决于要转换值的数据类型及其实际值。如表 3-1 所示给出了各种数据类型及其对象的转换规则。

表 3-1 Boolean 类型转换规则

数据类型	对应 true 值	对应 false 值
boolean	true	false
string	非空字符串	空字符串
number	非零数值	0 或 NaN
object	任何对象	null
undefined	不适用	undefined

这些转换规则对理解流控制语句(如 if 语句)自动执行相应的 Boolean 转换非常重要，由于存在这种自动执行的 Boolean 转换，因此确切地知道在流控制语句中使用的是什么变量至关重要。

4. Number类型

JavaScript 不是类型语言。与许多其他编程语言不同，JavaScript 不定义不同类型的数

值，比如整数，以及短、长、浮点数等。其数字类型有以下几个特点：

- JavaScript 中的所有数值都存储为根为 10 的 64 位(8 比特)浮点数。
- 整数(不使用小数点或指数计数法)最多为 15 位。
- 小数的最大位数是 17，但是浮点运算并不总是 100%准确。
- 如果前缀为 0，则 JavaScript 会把数值常量解释为八进制数，如果前缀为 0 和 x，则解释为十六进制数。

Number 类型用来表示整数和浮点数值，还有一种特殊的数值，即 NaN(非数值，Not a Number)。这个数值用于表示一个本来要返回数值的操作数未返回数值的情况(这样就不会抛出错误了)。

例如，在其他编程语言中，任何数值除以 0 都会导致错误，从而停止代码执行。但在 JavaScript 中，任何数值除以 0 会返回 NaN，因此不会影响其他代码的执行。

NaN 本身有两个非同寻常的特点，如下所示：

- 任何涉及 NaN 的操作(例如 NaN/10)都会返回 NaN，这个特点在多步计算中有可能导致问题。
- NaN 与任何值都不相等，包括 NaN 本身。

JavaScript 中有一个 isNaN()函数，这个函数接收一个参数，确定这个参数是否“不是数值”，该参数可以是任何类型。JavaScript 中有 3 个函数可以把非数值转换为数值：Number()、parseInt()和 parseFloat()。第一个函数，即转型函数 Number()可以用于任何数据类型，而另外两个函数则专门用于把字符串转换成数值。这 3 个函数对于同样的输入，会返回不同的结果。

Number()函数的转换规则如下：

- 如果是 Boolean 值，true 和 false 将分别被替换为 1 和 0。
- 如果是数值，只是简单地传入和返回。
- 如果是 null 值，返回 0。
- 如果是 undefined，返回 NaN。
- 如果是字符串，则需要根据字符串所含字符来确定。

如果是字符串，则遵循下列规则：

- 如果字符串中只包含数值，则将其转换为十进制数值，即 1 会变成 1，123 会变成 123，而 011 会变成 11(前面的 0 被忽略)。
- 如果字符串中包含有效的浮点格式，如 1.1，则将其转换为对应的浮点数(同样，也会忽略前面的 0)。
- 如果字符串中包含有效的十六进制格式，例如 0xf，则将其转换为相同大小的十进制整数值。
- 如果字符串是空的，则将其转换为 0。如果字符串中包含除了上述格式之外的字符，则将其转换为 NaN。
- 如果是对象，则调用对象的 valueOf()函数，然后依照前面的规则转换返回的值。
- 如果转换的结果是 NaN，则调用对象的 toString()函数，然后再依次按照前面的规则转换返回的字符串值。

由于 Number()函数在转换字符串时比较复杂而且不够合理，因此在处理整数的时候更常用的是 parseInt()函数。

parseInt()函数在转换字符串时，更多的是看其是否符合数值模式。它会忽略字符串前面的空格，直至找到第一个非空格字符。具体规则如下：

- 如果第一个字符串不是数字字符或者负号，parseInt()会返回 NaN。也就是说，用 parseInt()转换空字符串会返回 NaN。
- 如果第一个字符是数字字符，prarseInt()会继续解析第二个字符，直到解析完所有后续字符或者遇到了一个非数字字符。例如，1234blue 会被转换为 1234，22.5 会被转换为 22，因为小数点并不是有效的数字字符。
- 如果字符串中的第一个字符是数字字符，parseInt()也能够识别出各种整数格式(即十进制、八进制、十六进制)。

5. String类型

String 类型用于表示由零或多个 16 位 Unicode 字符组成的字符序列，即字符串。字符串可以由单引号或双引号表示。任何字符串的长度都可以通过访问其 length 属性取得。

要把一个值转换为一个字符串，有两种方式。第一种是使用 toString()函数；第二种是使用 String()函数。

(1) toString()函数

数值、布尔值、对象和字符串值都有 toString()函数。但 null 和 undefined 值没有这个函数。多数情况下，调用 toString()函数不必传递参数。但是，在调用数值的 toString()函数时，可以传递一个参数，即输出数值的基数。通过指定不同的基数，toString()函数会改变输出的值。

(2) String()函数

在不知道要转换的值是不是 null 或 undefined 的情况下，还可以使用转型函数 String()，这个函数能够将任何类型的值转换为字符串。String()函数遵循下列转换规则：

- 如果值有 toString()函数，则调用该函数(没有参数)并返回相应的结果。
- 如果值是 null，则返回 null。
- 如果值是 undefined，则返回 undefined。

6. Object类型

Object 类型是对象类型，对象其实就是一组数据和功能的集合。对象可以通过执行 new 操作符后跟要创建的对象类型的名称来创建。而创建 Object 类型的实例并为其添加属性和(或)函数，就可以创建自定义对象。如创建一个名为 obj 的对象，代码如下：

```
var obj = new Object();
```

Object 的每个实例都具有下列属性和函数。

- constructor：保存着用于创建当前对象的函数。
- hasOwnProperty(propertyName)：用于检查给定的属性在当前对象实例中(而不是在实例的原型中)是否存在。作为参数的属性名，propertyName 必须以字符串形式

指定。

- isPrototypeOf(object)：用于检查传入的对象是否是另一个对象的原型。
- propertyIsEnumerable(propertyName)：用于检查给定的属性是否能够使用 for-in 语句来枚举。
- toString()：返回对象的字符串表示。
- valueOf()：返回对象的字符串、数值或布尔值表示。通常与 toString()函数的返回值相同。

3.4 运 算 符

JavaScript 提供了丰富的运算功能，包括算术运算、关系运算、逻辑运算和连接运算。本节介绍 JavaScript 中的运算符。

1. 算术运算符

JavaScript 中的算术运算符有单目运算符和双目运算符。双目运算符包括+(加)、-(减)、*(乘)、/(除)、%(取余)、|(按位或)、&(按位与)、<<(左移)、>>(右移)等。单目运算符有-(取反)、~(取补)、++(递加 1)、--(递减 1)等。

2. 关系运算符

关系运算符又称比较运算，运算符包括<(小于)、<=(小于等于)、>(大于)、>=(大于等于)、==(等于)和!=(不等于)，此外还有===和!==。

关系运算的运算结果为布尔值，如果条件成立，则结果为 true，否则为 false。

对于等同运算符“==”的比较，当两个运算数的类型不同时，将它们转换成相同的类型。其转换方式及比较结果如下所示：

- 对于一个数值与一个字符串，字符串转换成数值之后，进行比较。
- true 转换为 1、false 转换为 0，进行比较。
- 对于一个对象、数组、函数与一个数值或字符串的比较，对象、数组、函数转换为原始类型的值，然后进行比较。先使用 valueOf()，如果不行就使用 toString()。

其他类型的组合比较，需要两个运算数类型相同，或转换成相同类型后比较。其比较结果如下所示：

- 对于两个字符串，如果同一位置上的字符相等，这两个字符串就相同。
- 对于两个数，两数相同就相同。如果一个是 NaN 或两个都是 NaN，则不相同。
- 如果两个都是 true，或者两个都是 false，则相同。
- 如果两个引用的是同一个对象、函数、数组，则相等，如果引用的不是同一个对象、函数、数组，则不相同，即使这两个对象、函数、数组可以转换成完全相等的原始值。
- 如果是两个 null，或者两个都是未定义的，那么它们相等。

在 JavaScript 中，“===”是全同运算符，只有当值相等，数据类型也相等时才成立。

全同运算符遵循等同运算符的比较规则，但是它不对运算数进行类型转换，当两个运算数的类型不同时，返回 false；只有当两个运算数的类型相同的时候，才遵循等同运算符的比较规则进行比较。

3. 逻辑运算符

逻辑运算符有：&&(逻辑与)、||(逻辑或)、!(取反，逻辑非)、^(逻辑异或)。

4. 字符串连接运算符

连接运算用于字符串操作，运算符为+(用于强制连接)，将两个或多个字符串连接为一个字符串。

5. 三目运算符

三目运算符“? :”是另一种比较类型，其格式为：

```
操作数? 表达式 1 : 表达式 2
```

对于三目运算符“? :”构成的表达式，其逻辑功能为：若操作数的结果为 true，则表述式的结果为表达式 1，否则为表达式 2。例如 max=(a>b)? a : b；该语句的功能就是将 a、b 中的较大的数赋给 max。

【例 3.4】

计算半径为 4 的圆形的周长和面积，使用三目运算符比较圆形的周长和面积并输出其中较大的值。代码如下：

```
<script>
   var area = 3.14 * 4 * 4;
   var len = 2 * 3.14 * 4;
   document.write("圆的面积值:" + area + "<br/>");
   document.write("圆的周长值:" + len + "<br/>");
   var max = (area > len) ? "圆的面积值比较大" : "圆的周长值比较大"
   document.write(max);
</script>
```

上述代码的执行结果如图 3-4 所示。

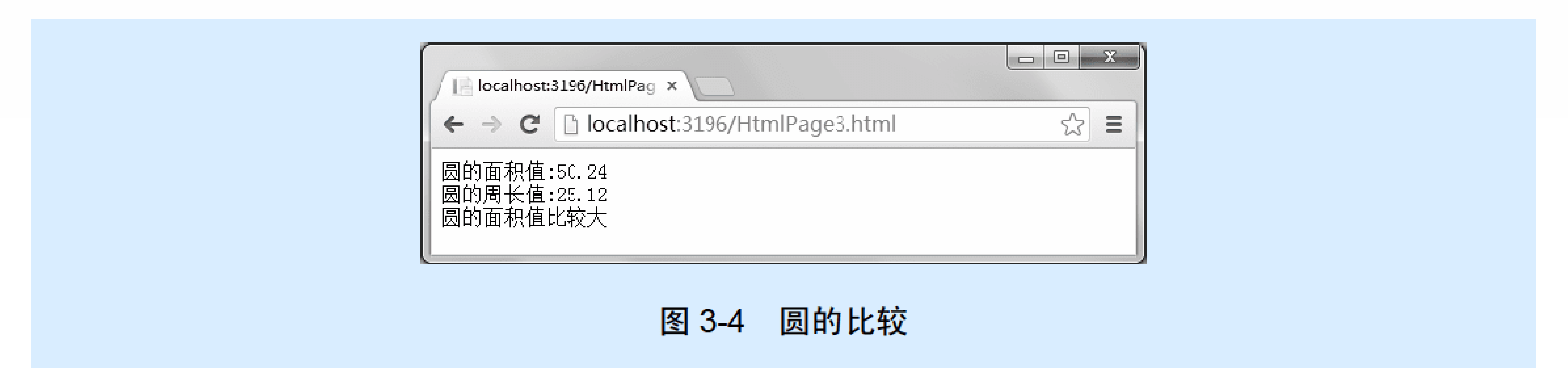


图 3-4 圆的比较

3.5 JavaScript 语句的类型

有其他编程语言基础的读者应该知道，程序开发语言都有着非顺序执行的语句，这种

语句包括选择语句、循环语句、跳转语句等。本节详细介绍 JavaScript 中的其他类型语句及其用法。

3.5.1 选择语句

选择语句用于执行满足指定条件的动作。如登录系统时的验证，当用户名密码正确时便可进入系统，但只要密码有误，就不能进入系统。这是一种选择，在不同的状态下系统接下来执行不同的操作。

JavaScript 提供了多种选择语句类型，以满足不同的程序需求，如下所示。

- if 语句：当满足条件时执行。
- if-else 语句：当满足条件时执行 if 后的语句，否则执行 else 后的语句。
- if-else-if-else-if 语句：当满足条件时执行 if 后的语句，否则满足第 2 个条件执行 else if 后的语句，否则满足第 3 个条件执行下一个 else if 后的语句。
- switch-case 语句：不同条件下执行不同语句。

1. if语句

if 语句执行时，首先判断条件表达式是否为真：条件为真则执行 if 语句下的语句块，结束条件语句；条件为假则直接结束条件语句块，执行 if 语句块后面的语句。其语法格式如下所示：

```
if (条件)
{
    //只有当条件为 true 时执行的代码
}
```

【例 3.5】

定义两个变量值 num1 和 num2，赋值为 12 和 16。比较两个值的大小，若 num1 大则输出“num1 比较大”；若 num2 大则输出“num2 比较大”。

代码如下：

```
<script>
    var num1=12,num2=16;

    if(num1>num2)
    {
        document.write("num1 比较大");
    }

    if(num2>num1)
    {
        document.write("num2 比较大");
    }
</script>
```

上述代码的执行结果如图 3-5 所示。可以看出，上述代码没有执行 if(num1>num2)后面的语句，而是执行了 if(num2>num1)后面的语句。

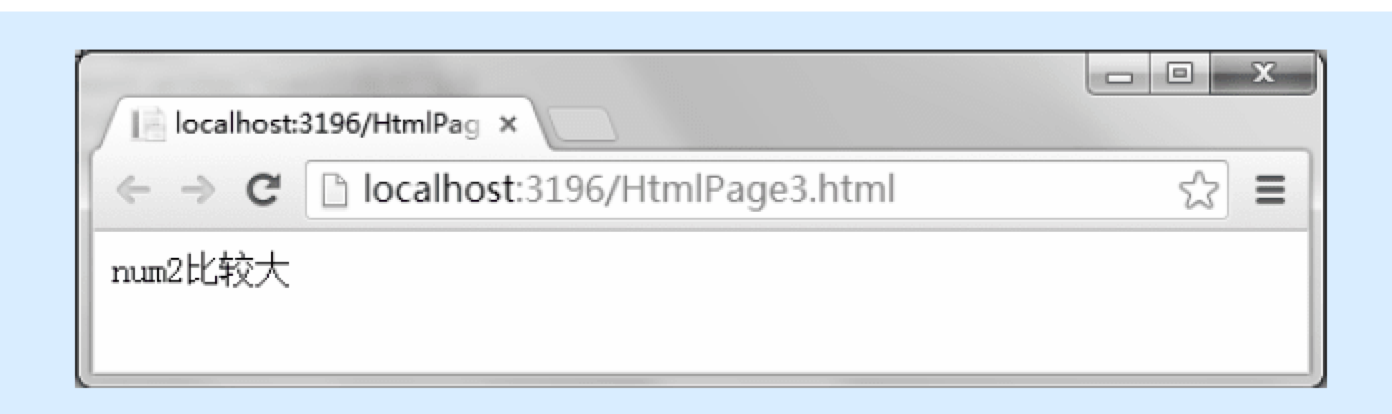


图 3-5 if语句的执行效果

2. if-else语句

if-else 语句在 if 语句的基础上添加了当条件不满足时进行的操作。条件的成立只有两种可能，即成立和不成立。if-else 语句在条件表达式成立时与 if 语句一样执行 if 后的语句块 1，并结束条件语句；条件表达式不成立时，执行 else 后的语句块 2，执行完成后结束条件语句。语法如下：

```
if (条件)
{
    //当条件为 true 时执行的代码
}
else
{
    //当条件不为 true 时执行的代码
}
```

【例 3.6】

修改例 3.5 中的代码为：比较两个值的大小，若 num1 大则输出“num1 比较大”；否则输出“num2 比较大”。

代码如下：

```
<script>
    var num1 = 12, num2 = 16;

    if (num1>num2)
    {
        document.write("num1 比较大");
    }
    else
    {
        document.write("num2 比较大");
    }
</script>
```

执行上述代码，其效果如图 3-5 所示。与例 3.5 的执行效果完全一样。由图 3-5 可以看出，上述代码没有执行 if(num1>num2)后面的语句，而是执行了 else 后面的语句。

3. if-else-if-else

if-else-if-else 语句相对复杂，它提供了多个条件来筛选数据，将数据依次分类排除。程序流程如图 3-6 所示。

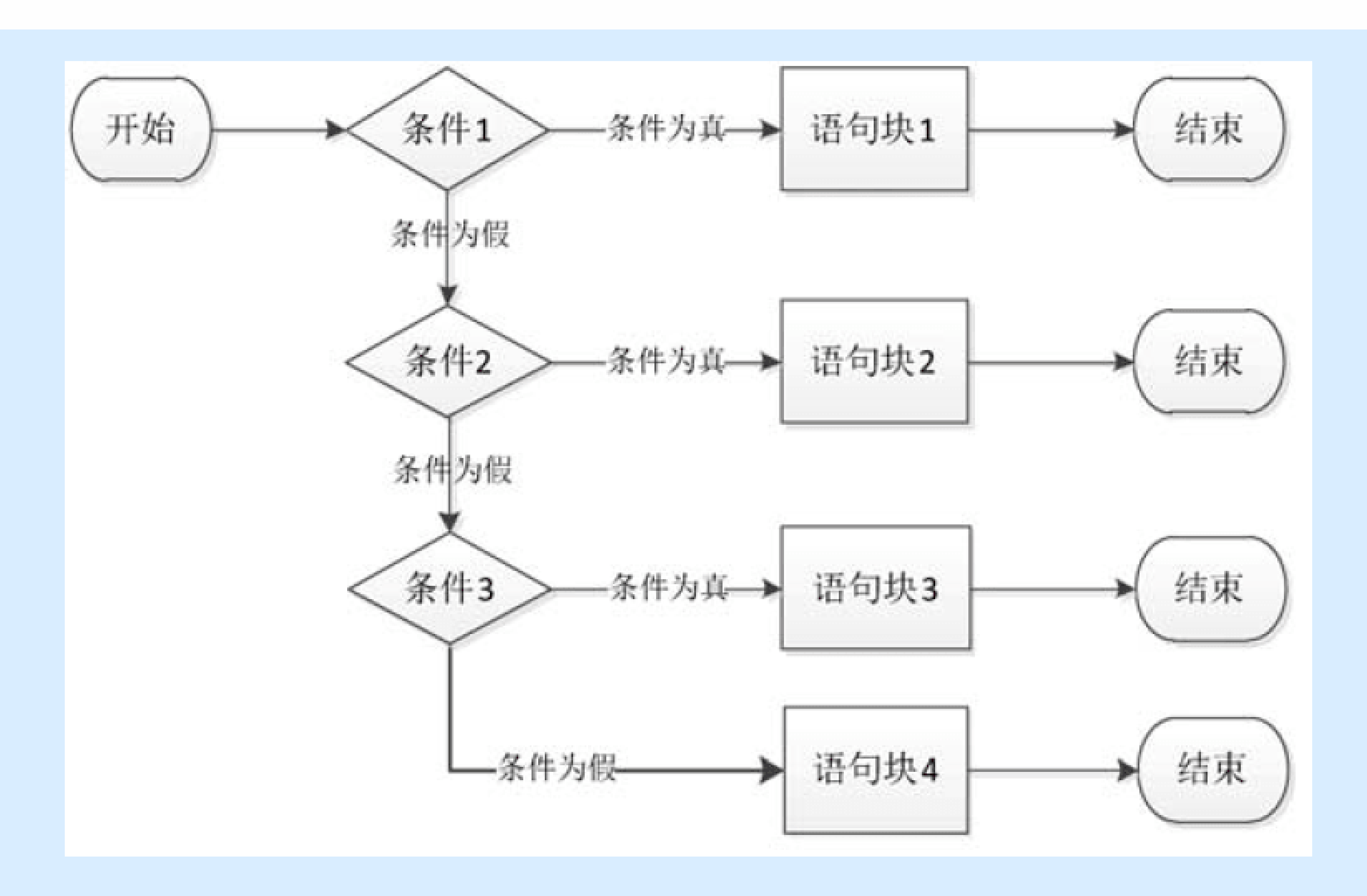


图 3-6　if-else-if-else语句的流程

图 3-6 中，if-else-if-else 语句在程序进入语句时，首先判定第一个 if 下的条件 1：

- 条件 1 成立，执行语句块 1 并结束条件语句。
- 条件 1 不成立，判断条件 2，条件 2 成立，执行语句块 2 并结束条件语句。
- 条件 2 不成立，判断条件 3，条件 3 成立，执行语句块 3 并结束条件语句。
- 条件 3 不成立，执行语句块 4 并结束条件语句。

图中只有 3 个条件和一个 else 语句。在 if-else-if-else 语句中，条件可以是任意多个，但 else 语句小于等于 1 个。即 else 语句可以不要，也可以要，要的话只能有 1 个，因为条件只有成立和不成立两种结果。其语法结构如下所示：

```
if (条件 1)
{
    //当条件 1 为 true 时执行的代码
}
else if (条件 2)
{
    //当条件 2 为 true 时执行的代码
}
else
{
    //当条件 1 和条件 2 都不为 true 时执行的代码
}
```

【例 3.7】

中国的大多数地区，冬季分布在 12 月份、1 月份和 2 月份；春季分布在 3~5 月份。创建月份变量，并根据月份判断所处的季节，代码如下：

```
<script>
    var num1 = 4;
    if (num1 == 12 || num1 == 1 || num1 == 2)
    {
        document.write("冬天");
    }
```

```
    else if (num1>2 && num1<6)
    {
        document.write("春天");
    }
    else if (num1>5 && num1<9)
    {
        document.write("夏天");
    }
    else
    {
        document.write("秋天");
    }
</script>
```

上述代码中，由于 12 月份、1 月份和 2 月份并不连续，因此需要使用“或”逻辑运算符来连接；剩下的月份使用“与”逻辑运算符链接。根据月份判断并输出当前季节，其执行结果为“春天”。由执行结果可以看出，程序在执行时跳过了不符合条件的语句，而直接执行 else if (num1>2 && num1<6)下的语句。

4. switch语句

switch 语句的完整形式为 switch ... case ... default。switch 语句与 if-else-if 语句用法相似，但 switch 语句中使用的条件只能是确定的值，即条件表达式等于某个常量，不能使用范围。switch 语句的流程如图 3-7 所示。

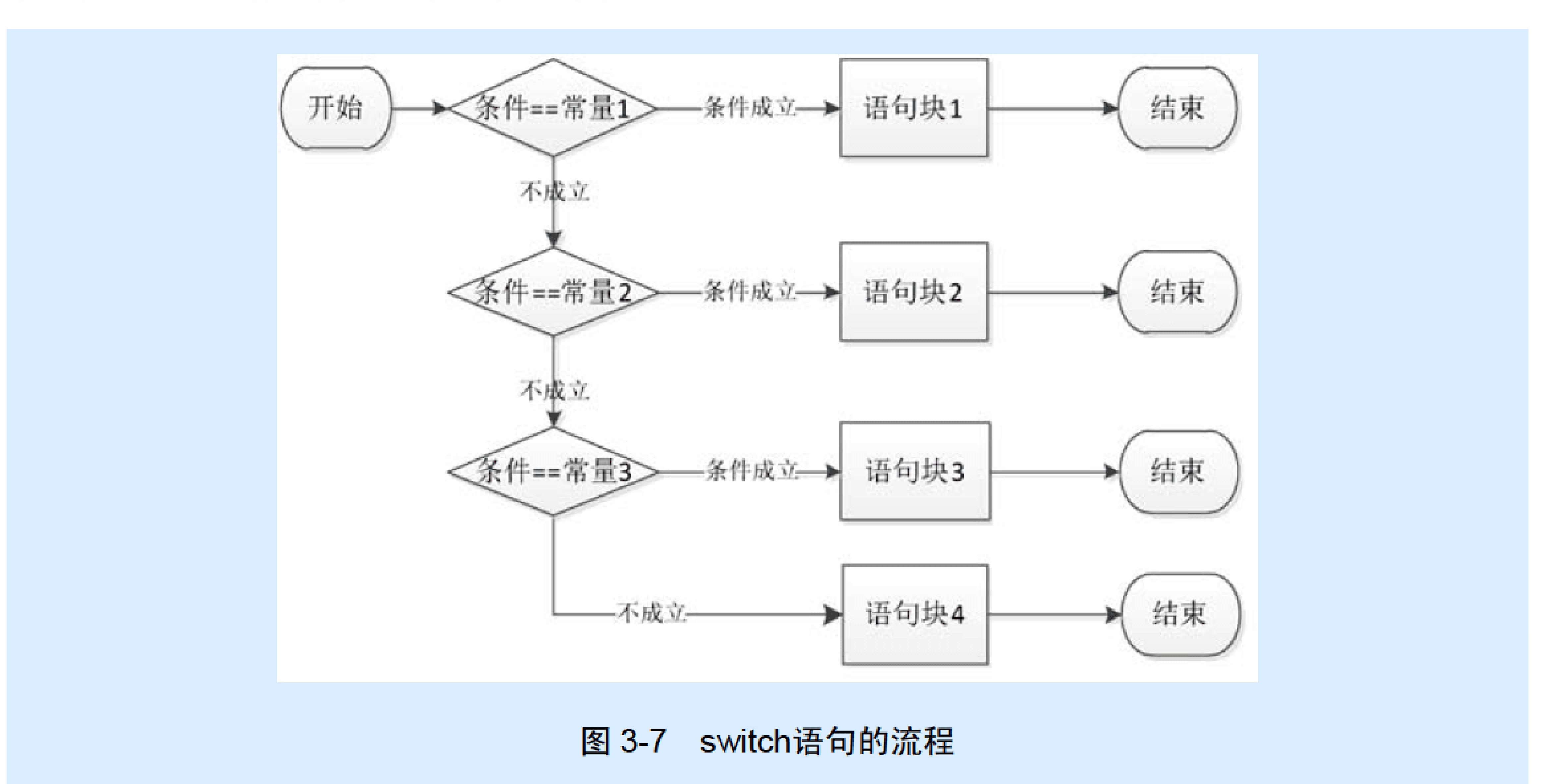


图 3-7　switch语句的流程

图 3-7 中，switch 语句在程序进入语句时，首先判定第一个常量 1 是否与条件相等。常量可以是具体数值，也可以是表达式。

- 条件与常量 1 相等：执行语句块 1 并结束条件语句。
- 条件与常量 2 相等：执行语句块 2 并结束条件语句。
- 条件与常量 3 相等：执行语句块 3 并结束条件语句。
- 条件与三个常量都不相等：执行语句块 4 并结束条件语句。

图 3-7 中只有 3 个条件表达式和一个 default 语句。default 语句表示剩余的情况下，与 else 类似。

与 if-else-if 语句一样，条件常量可以是任意多个，default 语句可以不要，也可以要，要的话只能有 1 个，因为条件只有成立和不成立两种结果。

switch 语句的基本语法如下：

```
switch(n)
{
case 1:
  //执行语句块 1
  break;
case 2:
  //执行语句块 2
  break;
default:
  //与 case 1 和 case 2 不同时执行的语句
}
```

【例 3.8】

一周有 7 天，创建变量，为其指定 1~7 之间的数字，并通过 switch 语句检测变量的值，输出对应的一周中的星期几，如数字 1 对应“星期一”。代码如下：

```
<script>
    var week = 4;
    switch (week) {
        case 1:
            document.write("星期一");
            break;
        case 2:
            document.write("星期二");
            break;
        ... //省略部分代码
        case 6:
            document.write("星期六");
            break;
        default:
            document.write("星期天");
            break;
    }
</script>
```

上述代码省略了变量为 3、4、5 的语句，可参考变量为 2 时的代码。上述语句除了 switch 语句的构成，还有 break 关键字，该关键字表示跳出当前的选择语句，其执行结果为“星期四”。

若不使用 break 关键字，上述代码将输出“星期四、星期五、星期六、星期天”，即把满足条件后所有的语句都执行一次。break 属于跳转语句，将在本章 3.5.3 节介绍。

3.5.2 循环语句

循环语句用于重复执行特定语句块，直到循环终止条件成立或遇到跳转语句。程序中

经常需要将一些语句重复执行，如果使用基本语句顺序地执行，将使开发人员重复工作，影响效率。

例如，对于 1+2+3+…+100，使用顺序语句需要将 100 个数相加；若加至 1000、10000 或更大的数，使得数据量加大，容易出错，不便管理。

使用循环语句可以简化这种过程，将指定语句或语句块根据条件重复执行。循环语句分为以下两种：

- for 循环重复执行一个语句或语句块，但在每次重复前，将会验证循环条件是否仍然成立。
- while 循环语句指定在特定条件下重复执行一个语句或语句块。

1. for循环

for 循环在重复执行的语句块之前加入循环执行条件，循环条件通常用来限制循环次数，执行流程如图 3-8 所示。

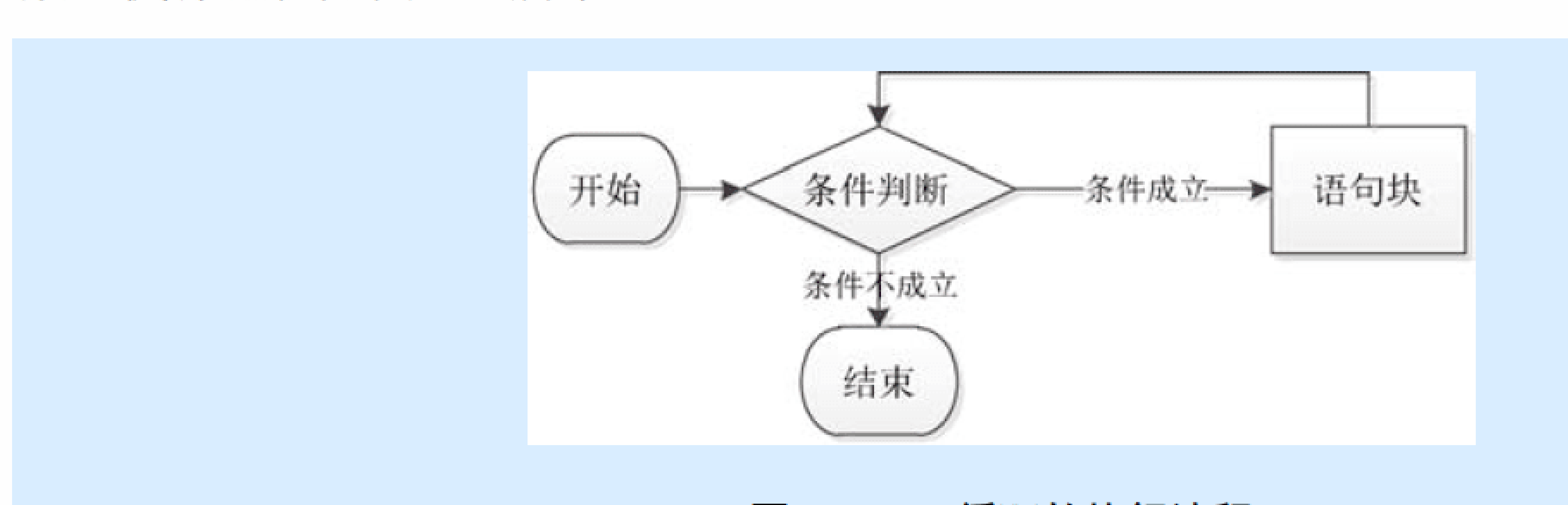


图 3-8　for循环的执行流程

如图 3-8 所示，从开始进入，判断循环条件是否成立：若成立，执行语句块，并重新判断循环条件是否成立；若不成立，结束这个循环。语法格式如下：

```
for(<初始化>; <条件表达式>; <增量>) {
    //语句块
}
```

for 语句执行括号里面语句的顺序如下。

(1) 首先是初始化的语句，如 var num=0；若 for 循环之前已经初始化，可以省略初始化表达式，但不能省略分号。

(2) 接着是条件表达式，如 num<5；表达式决定了该循环将在何时终止。表达式可以省略，但省略条件表达式，该循环将成为无限死循环。

(3) 最后是增量，通常是针对循环中初始化变量的增量，如 num++；增量与初始值和表达式共同决定了循环执行的次数。增量可以省略，但省略的增量将导致循环无法达到条件表达式的终止，因此需要在循环的语句块中修改变量值。

增量表达式后不需要分号，因 for 语句()内的 3 个表达式均可以省略，表达式间的分号不能省略，因此有以下空循环语句：

```
for (;;)
{
}
```

【例 3.9】

定义整型变量 num，计算 1+2+3+4+...+1000 的数值并赋值给 num，输出 num 的值。使用 for 循环语句如下：

```
<script>
    var num = 0;
    for (var x=1; x<=1000; x++) {
        num = num + x;
    }
    document.write(num);
</script>
```

上述代码的执行结果为“500500”。

2. while语句

while 语句在条件表达式判定之后执行循环，与 for 循环的执行顺序一样。不同的是语句格式和适用范围。

while 的使用比较灵活，甚至在某些情况下能替代条件语句和跳转语句。while 循环的流程如图 3-9 所示。

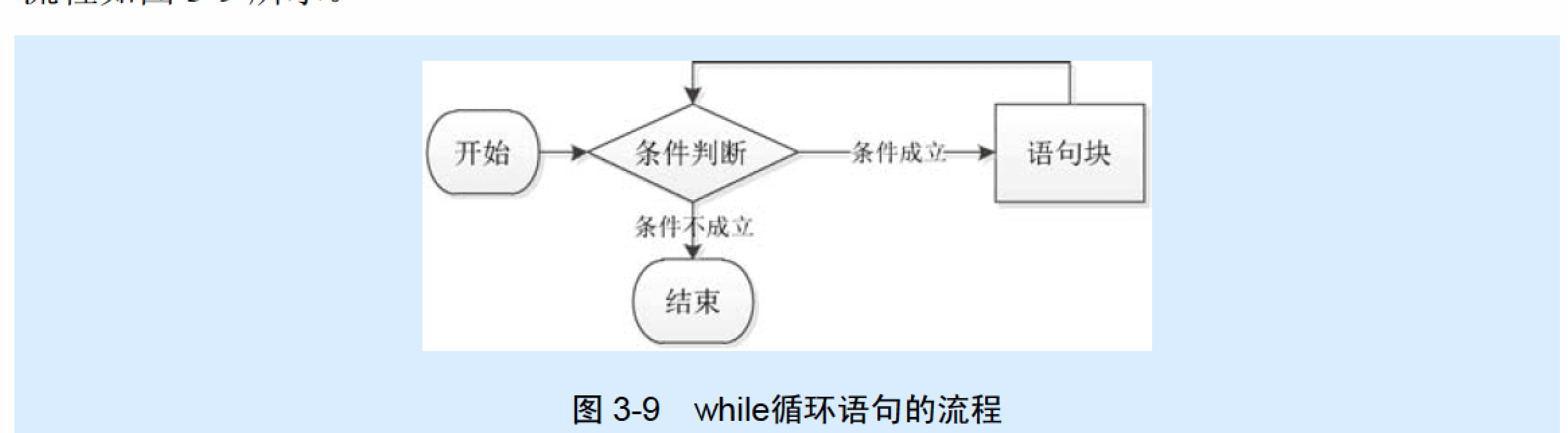


图 3-9 while循环语句的流程

图 3-9 中，执行至 while 语句时首先判断循环条件是否成立：若成立，执行语句块，并重新判断循环条件是否成立；若不成立，结束这个循环。语法格式如下：

```
while(条件表达式) {
    //语句块
}
```

从 while 的使用格式可以看出，while 的使用与 for 很接近，满足条件表达式即执行 while 语句块，否则结束循环。

while 后的括号()只能使用一个条件表达语句，若在循环中不改变条件表达式中的变量值，循环将无限进行下去，因此循环语句块中包含改变变量值的语句。

for 主要控制循环的次数，而对于不确定次数的循环，使用 while 比较合适。如同样计算 1~1000 这 1000 个数的和，使用 while 的情况如例 3.10 所示。

【例 3.10】

定义整型变量 num，计算 1+2+3+4+...+1000 的数值并赋值给 num，输出 num 的值。使用 while 循环语句如下：

```
<script>
    var i=0,num=0;
    while(i<=1000)
    {
        num = num + i;
        i++;
    }
    document.write(num);
</script>
```

运行上述代码，其结果与例 3.9 的结果一样。代码中，i 变量在循环中改变了数值，否则循环将永远进行下去，document.write(num);语句将无法被执行。

3.5.3 跳转语句

例 3.8 中使用的 break 语句即为跳转语句，若不使用该语句，程序运行时在跳过输出星期一到星期三之后，将执行输出星期四以及之后的所有输出函数。JavaScript 中的跳转语句有两种：break 语句和 continue 语句。

1. break语句

break 语句只能用在循环或 switch 中。循环有多种，任意一种循环都可以使用 break 跳出。break 有以下两种用法：

- 用在 switch 语句的 case 语句之后，结束 switch 语句块，执行 switch{}后的语句。
- 用在循环体，结束循环，执行循环{}后的语句。

2. continue语句

continue 语句是跳转语句的一种，用在循环中可以加速循环，但不能结束循环。continue 语句与 break 的不同之处在于：

- continue 语句不能用于选择语句。
- continue 语句在循环中不是跳出循环块，而是结束当前循环，进入下一个循环，忽略当前循环的剩余语句。

【例 3.11】

输出 30 以内的整数，在输出时跳过 5 的倍数，并在达到 5 的倍数时换行，使用循环语句和选择语句的嵌套，代码如下：

```
<script>
    for (var i=1; i<30; i++) {
        if (i%5 == 0) {
            document.write("<br/>");
            continue;
        }
        if (i < 10) {
            document.write(" " + i + " ");
        }
        else {
            document.write(i);
            document.write(" ");
```

```
        }
    }
</script>
```

上述代码的执行效果如图 3-10 所示。程序在执行到 5 的倍数时中止，但该循环并没有被跳出，而是接着执行后面的循环。

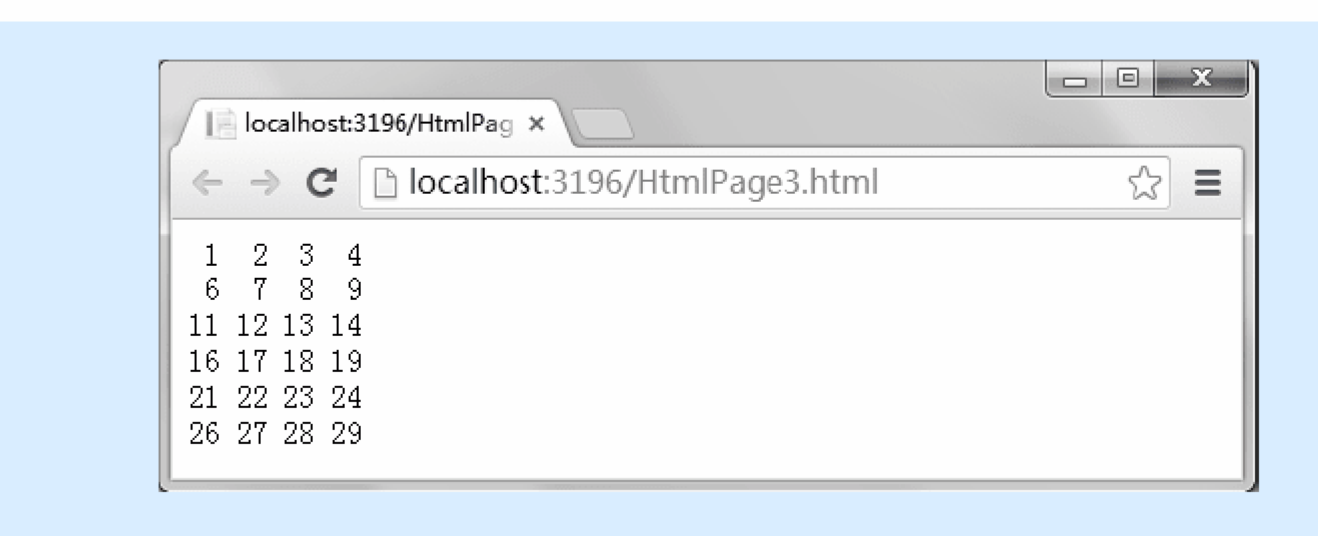


图 3-10　使用continue的执行效果

将代码中的 continue 改为 break，其执行效果如图 3-11 所示。

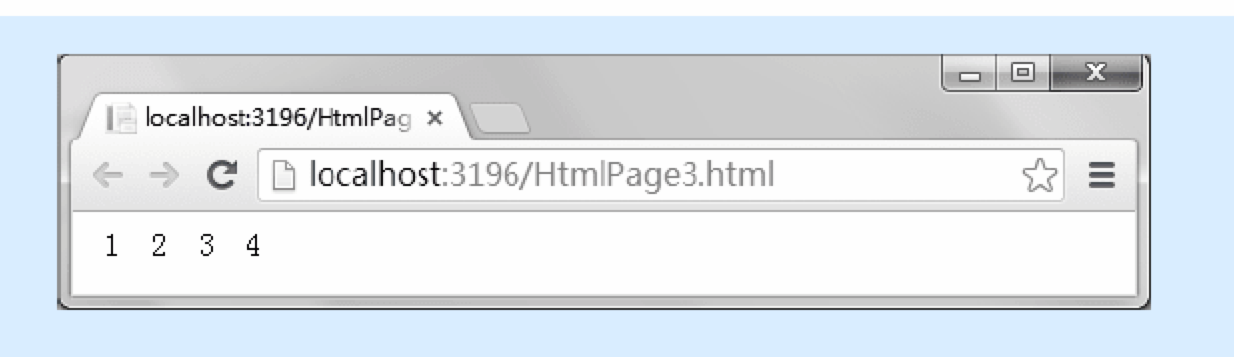


图 3-11　使用break的执行效果

由图 3-11 可以看出，程序在执行到 5 的倍数时彻底中断，该循环被直接跳出，没有接着执行 i 为 6 的情况。

3.5.4 异常处理语句

程序中不可避免地存在无法预知的反常情况，这种反常称为异常。JavaScript 为处理在程序执行期间可能出现的异常提供了内置支持，由正常控制流之外的代码处理。

本节介绍 JavaScript 内置的异常处理，包括使用 throw 抛出异常；使用 try 尝试执行代码；并在失败时使用 catch 处理异常。

当错误发生时，JavaScript 引擎通常会停止，并生成一个错误消息。把 throw、try 和 catch 一起使用，可以控制程序流，并生成自定义的错误消息。throw、try 和 catch 有以下几个特点：

- try 语句允许定义在执行时进行错误测试的代码块。
- catch 语句允许定义当 try 代码块发生错误时所执行的代码块。
- JavaScript 语句 try 和 catch 是成对出现的。
- throw 语句用于发出在程序执行期间出现异常的信号。

throw 语句将引发异常，当异常引发时，程序查找处理此异常的 catch 语句。也可以用 throw 语句重新引发已捕获的异常。throw 只是用在程序中的一条语句，语法如下：

```
throw exception
```

上述代码中，异常可以是 JavaScript 字符串、数字、逻辑值或对象。

把 throw、try 和 catch 一起使用，语法如下：

```
try
{
    //在这里运行代码
    throw exception  //抛出异常
}
catch(err)
{
    //在这里处理错误
}
```

【例 3.12】

定义变量 x，判断该变量的值是否为空、是否不是数字或是否太大和太小，根据不同的情况抛出不同的异常，并处理该异常，代码如下：

```
<script>
    var x = "";

    try {
        if (x == "") { throw "值为空"; }
        if (isNaN(x)) { throw "不是数字"; }
        if (x > 10) { throw "太大"; }
        if (x < 5) { throw "太小"; }
    }
    catch (err) {
        document.write(err);
    }

</script>
```

上述代码的执行效果为“值为空”，该异常是当 x==“”时抛出的异常，被 catch 语句处理时输出。

3.6 对　　象

JavaScript 中的所有事物都是对象，包括：字符串、数字、数组、日期，等等。在 JavaScript 中，对象是拥有属性和函数的数据。

3.6.1 对象概述

有过其他编程语言基础的读者应该知道，在面向对象编程语言中有类和对象的概念。将有相同数据和操作的语句放在一个类中，通过对类的实例化来调用类里面的数据和操作。这些数据作为类的属性，方法作为类的函数，共同定义一系列相关的数据和操作。

如长方体是生活中的一个对象，该对象可以有长度、宽度和高度这些数据，有求表面积、体积、棱长等这些操作，将这些数据定义为长方体对象的属性，将这些操作定义为长

方体对象的函数，那么一切与长方体有关的操作就都可以直接调用这个对象了。

JavaScript 本身并不支持面向对象，它没有访问控制符，没有定义类的关键字 class，没有支持继承的 extend 或冒号，也没有用来支持虚函数的 virtual，不过，JavaScript 是一门灵活的语言，能够以其他形式完成类和对象的功能。

在 JavaScript 中对象是一种数据(变量)，拥有属性和函数。当声明一个 JavaScript 变量如“var txt=“Hello”;”时，实际上已经创建了一个 JavaScript 字符串对象。字符串对象拥有内建的属性 length。对于上面的字符串来说，length 的值是 5。字符串对象同时拥有若干个内建的函数。

在面向对象的语言中，属性和函数常被称为对象的成员。JavaScript 中几乎所有事务都是对象：字符串、数字、数组、日期、函数等。

对象的名称由用户定义，在创建时使用 new 关键字和 Object()函数，如声明一个名为 person 的对象，同时为其添加属性，代码如下：

```
var person = new Object();
person.name = "Bill";
person.age = 56;
```

创建新 JavaScript 对象时可以包含多个函数，对于已经存在的对象，可以添加属性和函数。对象中属性和函数的调用需要用“对象名.属性名”或“对象名.函数名”的方式来进行。如例 3.13 定义了一个圆形对象。

【例 3.13】

创建圆形对象，为其指定半径为 5、计算其周长和面积并定义为对象的属性，输出圆的周长和面积，代码如下：

```
<script>
    var round = new Object();
    round.r = 5;
    round.l = 2*3.14 * round.r;
    round.s = 3.14 * round.r * round.r;
    document.write("圆的周长: " + round.l + "<br/>");
    document.write("圆的面积: " + round.s);
</script>
```

上述代码的执行效果如图 3-12 所示。

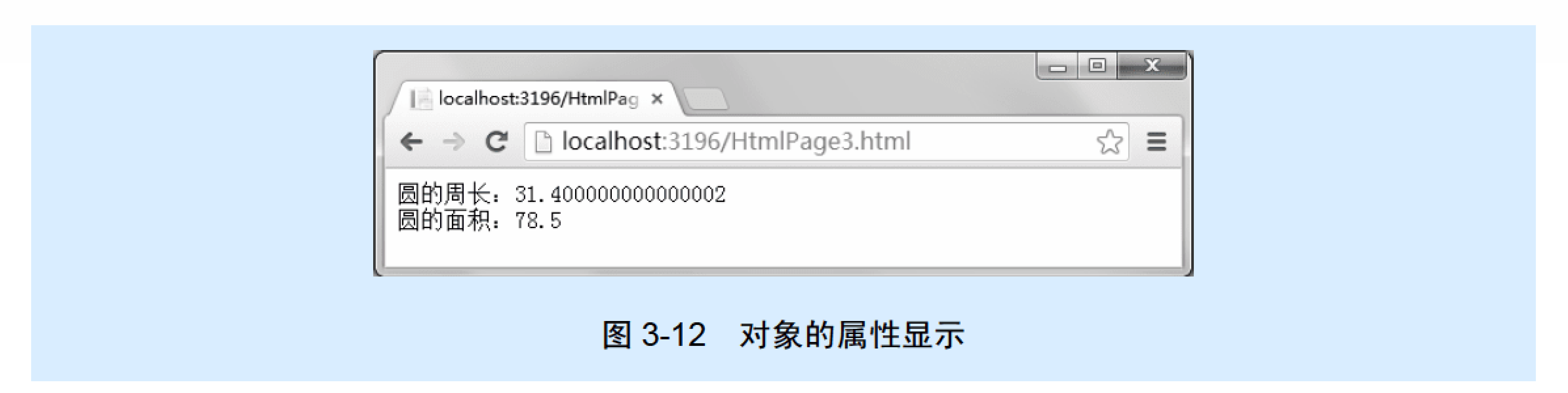


图 3-12　对象的属性显示

例 3.13 只是定义了一个对象中固定的一些数据；而若要在 JavaScript 中实现类一样的功能，能够拥有数据和操作，能够被封装和继承，需要使用函数的定义，这些将在本章 3.6.2 小节和 3.6.3 小节中介绍。

3.6.2 函数

函数在多种程序开发语言中都有使用，而且函数的定义和使用方法相似。函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块，在 JavaScript 中定义函数时使用 function 关键字，格式如下：

```
function functionname()
{
    //这里是要执行的代码
}
```

上述代码定义了函数 functionname()，当调用该函数时，会执行函数内的代码。函数定义之后，可以在某事件发生时直接调用函数(比如当用户单击按钮时)，也可以由 JavaScript 在任何位置进行调用。

提示：JavaScript 对大小写敏感。关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

在调用函数时可以向其传递值，这些值被称为参数。参数作为函数内部的变量来使用，定义一个函数的多个参数时，使用逗号隔开。参数在声明时不需要使用 var 关键字，如声明含有两个参数的函数 myFunction()，代码如下：

```
function myFunction(var1, var2)
{
   //这里是要执行的代码
}
```

变量和参数必须以一致的顺序出现。第一个变量就是第一个被传递的参数的给定的值，以此类推。

函数是可以有返回值的，返回值使用 return 语句来实现，在使用 return 语句时，函数会停止执行，并返回指定的值。

【例 3.14】

将例 3.13 中的对象定义为函数，将半径定义为函数的参数，函数返回值为圆的面积。定义之后引用该函数，代码如下：

```
<script>
   function round(r) {
      return 3.14 * r * r;
   }
   document.write("圆的面积：" + round(5));
</script>
```

上述代码首先定义了函数 round(r)，并在函数内以 r 为圆的半径计算圆的面积。接着在函数外部对函数进行调用，为函数的参数赋值 5 并输出，计算出圆的面积，如图 3-13 所示。

在函数的内部和外部都可以使用变量，但不同的是，函数内部的变量属于局部变量，而函数外部的变量使用全局变量。



图 3-13　圆的面积函数调用

(1)　局部 JavaScript 变量有如下特点：

- 在 JavaScript 函数内部声明的变量(使用 var)是局部变量，所以只能在函数内部访问它。该变量的作用域是局部的。
- 可以在不同的函数中使用名称相同的局部变量，因为只有声明过该变量的函数才能识别出该变量。
- JavaScript 变量的生命期从它们被声明的时间开始，只要函数运行完毕，本地变量就会被删除。

(2)　全局 JavaScript 变量有如下特点：

- 在函数外声明的变量是全局变量，网页上的所有脚本和函数都能访问它。
- 全局变量会在页面关闭后被删除。

3.6.3　构造函数

在 JavaScript 中可以对函数进行实例化，其效果相当于类的实例化。函数的实例化同样是使用 new 关键字，而且在函数中可以有数据，这些数据可以像类的属性一样使用，如例 3.15 所示。

【例 3.15】

构造圆的函数，有成员：半径 r、周长 l 和面积 s。在函数以外为函数创建对象，获取其周长和面积的值并输出，代码如下：

```
<script>

    function round() {
        this.r = 6;
        this.l = 3.14 * 2 * this.r;
        this.s = 3.14 * this.r * this.r;
    }

    var area = new round();

    document.write("圆的面积：" + area.s + "<br/>");
    document.write("圆的周长：" + area.l);

</script>
```

在上述代码中，函数的成员都使用 this 关键字来定义，表示该属性是共有的。若直接定义“var r = 6;”，那么该成员是私有的，在函数外部将无法访问到。上述代码的执行效果如图 3-14 所示。

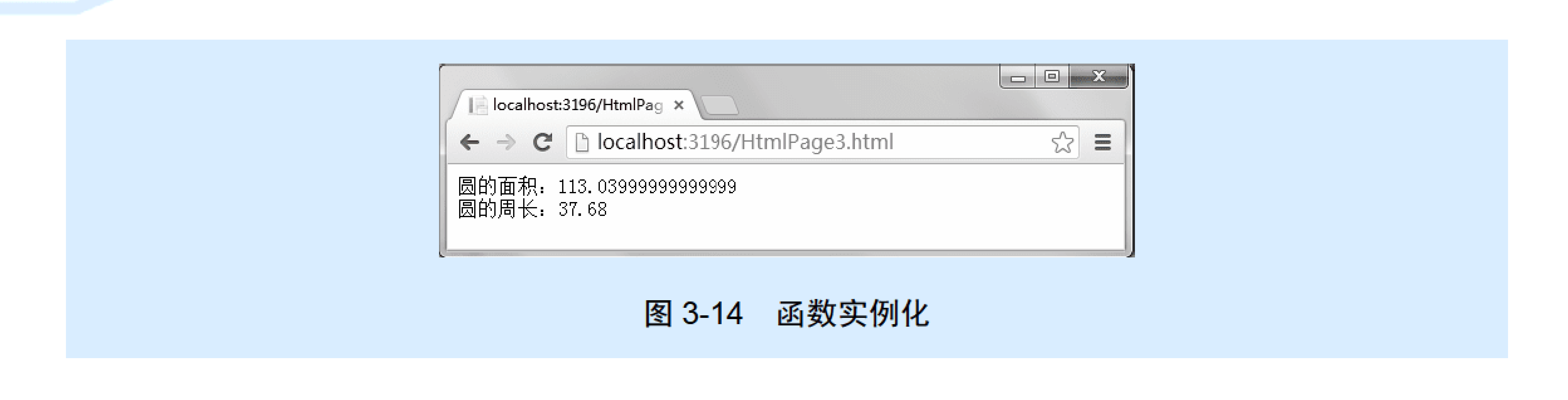


图 3-14　函数实例化

3.7 常 用 对 象

JavaScript 提供了内置的对象以实现特定的功能，其常用对象有数组对象、窗体对象和 DOM 对象等。本节详细介绍 JavaScript 内置对象的使用。

3.7.1 Array对象

Array 对象是 JavaScript 中的数组对象，实现数组的相关操作。数组允许在单个变量中存储多个值，其创建语法如下所示：

```
new Array();
new Array(size);
new Array(element0, element1, ..., elementn);
```

上述代码中，参数 size 是期望的数组元素个数。参数 element0...elementn 是参数列表。当使用这些参数来调用构造函数 Array()时，新创建的数组的元素就会被初始化为这些值。它的长度(元素数目)也会被设置为参数的个数。

如果调用构造函数 Array()时没有使用参数，那么返回的数组为空，元素数目为 0。

如果调用构造函数时只传递给它一个数字参数，该构造函数将返回具有指定个数、元素为 undefined 的数组。

当其他参数调用 Array()时，该构造函数将用参数指定的值初始化数组。

当把构造函数作为函数调用，不使用 new 运算符时，它的行为与使用 new 运算符调用它时的行为完全一样。

Array 对象有以下 3 个属性。

- constructor：返回对创建此对象的数组函数的引用。
- length：设置或返回数组中元素的数目。
- prototype：使开发人员有能力向对象添加属性和方法。

Array 对象有着对数组的操作，其所包括的方法如表 3-2 所示。

表 3-2　Array 对象的方法

方法名称	说　明
concat()	连接两个或更多的数组，并返回结果
join()	把数组的所有元素放入一个字符串元素，通过指定的分隔符进行分隔
pop()	删除并返回数组的最后一个元素

续表

方法名称	说　明
push()	向数组的末尾添加一个或更多元素，并返回新的长度
reverse()	颠倒数组中元素的顺序
shift()	删除并返回数组的第一个元素
slice()	从某个已有的数组返回选定的元素
sort()	对数组的元素进行排序
splice()	删除元素，并向数组添加新元素
toSource()	返回该对象的源代码
toString()	把数组转换为字符串，并返回结果
toLocaleString()	把数组转换为本地数组，并返回结果
unshift()	向数组的开头添加一个或更多元素，并返回新的长度
valueOf()	返回数组对象的原始值

3.7.2 Document对象

Document 对象使设计人员可以从脚本中对 HTML 页面中的元素进行访问。Document 对象是 Window 对象的一部分，可通过 window.document 属性对其进行访问。

Document 对象可以控制页面中的元素，也可以对多个元素统一处理。对多个元素统一处理需要使用集合，其所包含的集合如表 3-3 所示。

表 3-3　Document 对象的集合

集　合	说　明
all[]	提供对文档中所有 HTML 元素的访问
anchors[]	返回对文档中所有 Anchor 对象的引用
applets	返回对文档中所有 Applet 对象的引用
forms[]	返回对文档中所有 Form 对象的引用
images[]	返回对文档中所有 Image 对象的引用
links[]	返回对文档中所有 Area 和 Link 对象的引用

HTML Document 接口对 DOM Document 接口进行了扩展，定义 HTML 专用的属性和方法。

很多属性和方法都是 HTML Collection 对象拥有的，其中保存了对锚、表单、链接以及其他可脚本化元素的引用。其常用属性和方法如表 3-4 和表 3-5 所示。

表 3-4　Document 对象的属性

属性名称	说　明
body	提供对<body>元素的直接访问。对于定义了框架集的文档，该属性引用最外层的<frameset>

续表

属性名称	说　明
cookie	设置或返回与当前文档有关的所有 cookie
domain	返回当前文档的域名
lastModified	返回文档被最后修改的日期和时间
referrer	返回载入当前文档的 URL
title	返回当前文档的标题
URL	返回当前文档的 URL

表 3-5　Document 对象的方法

方法名称	描　述
close()	关闭用 document.open()方法打开的输出流，并显示选定的数据
getElementById()	返回对拥有指定 id 的第一个对象的引用
getElementsByName()	返回带有指定名称的对象集合
getElementsByTagName()	返回带有指定标签名的对象集合
open()	打开一个流，以收集来自任何 document.write()或 document.writeln()方法的输出
write()	向文档写 HTML 表达式或 JavaScript 代码
writeln()	等同于 write()方法，不同的是在每个表达式之后写一个换行符

write()方法值得注意，在文档载入和解析的时候，它允许一个脚本向文档中插入动态生成的内容。

3.7.3 HTML DOM Event对象

Event 对象监控事件的状态，其事件包括键盘事件和鼠标事件，事件的状态包括键盘按钮的按下、松开；鼠标的单击、双击等。

为这些事件定义脚本，可使页面与用户之间建立互动，页面根据用户的操作执行相应的脚本。事件通常与函数结合使用，函数不会在事件发生前被执行。

Event 对象的应用通常是对 Event 对象属性的应用，其属性定义了元素的操作事件，为元素的事件定义脚本，以实现元素的功能。其常用属性如表 3-6 所示。

表 3-6　Event 对象的常用属性

属性名称	说　明
onclick	当用户单击某个对象时调用的事件
ondblclick	当用户双击某个对象时调用的事件
onfocus	元素获得焦点
onkeydown	某个键盘按键被按下

续表

属性名称	说　明
onkeypress	某个键盘按键被按下并松开
onkeyup	某个键盘按键被松开
onload	一张页面或一幅图像完成加载
onmousedown	鼠标按钮被按下
onmousemove	鼠标被移动
onmouseout	鼠标从某元素移开
onmouseover	鼠标移到某元素之上
onmouseup	鼠标按键被松开
onreset	重置按钮被点击
onselect	文本被选中
onsubmit	确认按钮被点击
onunload	用户退出页面

3.7.4 Window对象

Window 对象表示一个浏览器窗口或一个框架。在客户端 JavaScript 中，Window 对象是全局对象，所有的表达式都在当前的环境中计算。也就是说，要引用当前窗口，根本不需要特殊的语法，可以把那个窗口的属性作为全局变量来使用。

例如，可以只写document，而不必写window.document。同样，可以把当前窗口对象的方法当作函数来使用，如只写alert()，而不必写window.alert()。Window对象的常用方法如表 3-7 所示。

表 3-7　Window 对象的方法

方法名称	说　明
alert()	显示带有一段消息和一个确认按钮的警告框
blur()	把键盘焦点从顶层窗口移开
clearInterval()	取消由 setInterval()设置的 timeout
clearTimeout()	取消由 setTimeout()方法设置的 timeout
close()	关闭浏览器窗口
confirm()	显示带有一段消息以及确认按钮和取消按钮的对话框
createPopup()	创建一个 pop-up 窗口
focus()	把键盘焦点给予一个窗口
moveBy()	可相对窗口的当前坐标把它移动指定的像素
moveTo()	把窗口的左上角移动到一个指定的坐标
open()	打开一个新的浏览器窗口或查找一个已命名的窗口
print()	打印当前窗口的内容

续表

方法名称	说　明
prompt()	显示可提示用户输入的对话框
resizeBy()	按照指定的像素调整窗口的大小
resizeTo()	把窗口的大小调整到指定的宽度和高度
scrollBy()	按照指定的像素值来滚动内容
scrollTo()	把内容滚动到指定的坐标
setInterval()	按照指定的周期(以毫秒计)来调用函数或计算表达式
setTimeout()	在指定的毫秒数后调用函数或计算表达式

除了表 3-7 所列出的方法，Window 对象还实现了核心 JavaScript 所定义的所有全局属性和方法。

Window对象的window属性和self属性引用的都是它自己。当明确地引用当前窗口，而不仅仅是隐式地引用它时，可以使用这两个属性。除了这两个属性之外，parent属性、top属性以及frame[]数组都引用了与当前Window对象相关的其他Window对象。

新的顶层浏览器窗口由方法window.open()创建。当调用该方法时，应把open()调用的返回值存储在一个变量中，然后使用那个变量来引用新窗口。新窗口的opener属性反过来引用了打开它的那个窗口。

一般来说，Window对象的方法都是对浏览器窗口或框架进行某种操作。而alert()方法、confirm()方法和prompt方法则不同，它们通过简单的对话框与用户进行交互。

3.8　实战——长方体几何计算

本章全面讲述了 JavaScript 的基础知识，包括 JavaScript 中的语法规则、语句、变量、运算符、对象和函数等。本节结合本章内容，创建长方体函数并对其进行实例化。具体要求如下：

- 创建长方体计算函数，有长方体的长、宽和高这 3 个参数。
- 在函数中计算获取长方体的体积属性值和表面积属性值。
- 创建长 4、宽 3、高 2 的长方体和长宽高均为 3 的正方体。
- 计算这两个长方体的面积和体积并输出。
- 判断长方体和正方体的体积大小并输出。

实现上述要求，步骤如下。

步骤 01　创建长方体计算函数，有长方体的长、宽和高这 3 个参数，在函数中计算获取长方体的体积属性值和表面积属性值，函数代码如下：

```
<script>
   function boxes(l, w, h) {
      this.l = l;
      this.w = w;
      this.h = h;
```

```
        this.area = 2 * (l * w + l * h + w * h);
        this.volume = l * h * w;
    }
</script>
```

步骤 02 创建长 4、宽 3、高 2 的长方体和长宽高均为 3 的正方体，计算这两个长方体的面积和体积并输出，代码如下：

```
var box1 = new boxes(2, 3, 4);
document.write("长方体表面积： " + box1.area + "<br/>");
document.write("长方体体积： " + box1.volume + "<br/>");
var box2 = new boxes(3, 3, 3);
document.write("正方体表面积： " + box2.area + "<br/>");
document.write("正方体体积： " + box2.volume + "<br/>");
```

步骤 03 判断长方体和正方体的体积大小并输出，代码如下：

```
if (box1.area > box2.area)
{
    document.write("长方体体积较大");
}
else if (box1.area == box2.area)
{
    document.write("体积一样大");
}
else
{
    document.write("正方体体积较大");
}
```

步骤 04 运行上述代码，执行效果如图 3-15 所示。

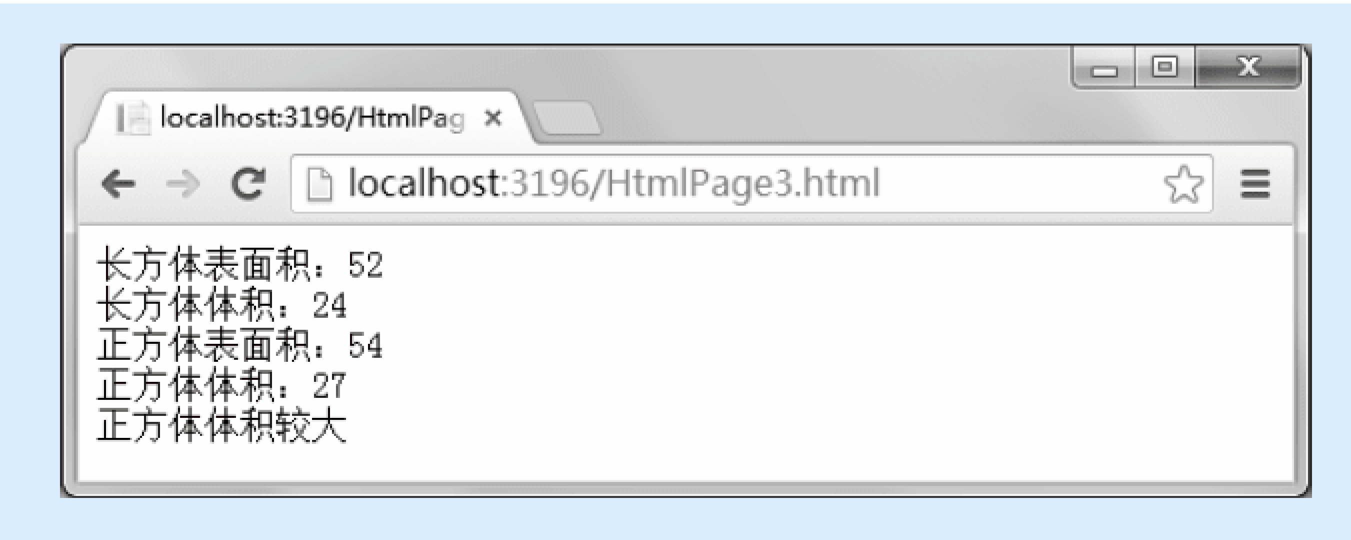


图 3-15　长方体几何计算的执行效果

3.9 本 章 习 题

1. 填空题

(1) HTML 中的脚本必须位于__________标记之间。

(2) HTML5 中默认使用__________脚本语言。

(3) JavaScript 中输出不换行的文本的函数是__________。

(4) JavaScript 单行注释使用__________符号。

(5) JavaScript 多行注释使用＿＿＿＿＿＿符号和"*/"符号。

(6) 数字类型中的 0 对于 Boolean 类型中的＿＿＿＿＿＿。

2. 选择题

(1) 以下关于 JavaScript 的说法正确的是＿＿＿＿＿＿。

A. JavaScript 中分号和换行都可以作为 JavaScript 语句的结束

B. JavaScript 语句不区分大小写

C. JavaScript 中的字符使用单引号，字符串使用双引号

D. JavaScript 中每一条语句都以分号结尾

(2) 下列不属于 JavaScript 选择语句的是＿＿＿＿＿＿。

A. if 语句　　B. if-else 语句

C. continue 语句　　D. switch-case 语句

(3) 下列保留字与选择语句无关的是＿＿＿＿＿＿。

A. if　　B. case　　C. break　　D. continue

(4) 下列语句与循环语句无关的是＿＿＿＿＿＿。

A. for 语句　　B. do-while 语句　　C. break 语句　　D. while 语句

(5) 下列关于变量的说法正确的是＿＿＿＿＿＿。

A. 变量不可以以特殊符号开头，如$符号

B. JavaScript 支持多种数据类型，因此在声明变量时需要声明其数据类型

C. 变量在没有赋值的情况下值为 undefined

D. 变量在没有赋值的情况下值为 null

(6) 下列不属于逻辑运算符的是＿＿＿＿＿＿。

A. &&　　B. ||　　C. !　　D. &

3. 上机练习

使用一种符号，如@、#、*或$等，输出一个直角梯形。要求在梯形每一行的中间位置使用另一种符号，达到如图 3-16 所示的效果。

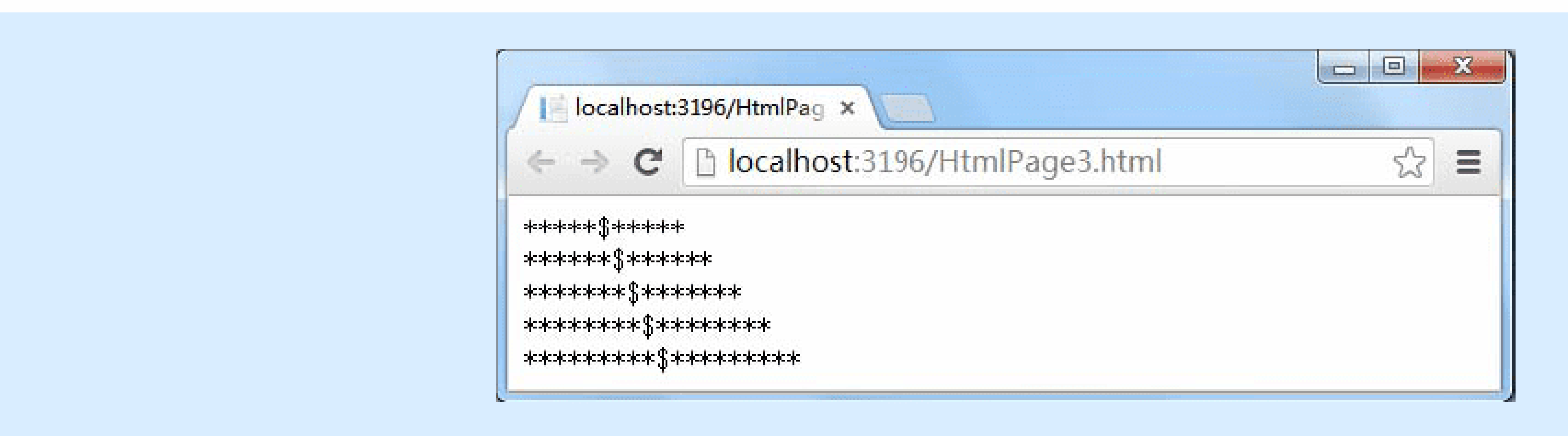


图 3-16　上机练习的效果





第4章 网页设计实战案例

通过前面章节的学习，读者已经掌握了设计网页的所有知识，包括制作网页内容的 HTML、控制网页效果的 CSS 以及实现网页特效的 JavaScript。

本章将为读者介绍实际网页设计时所需掌握的各种技能。首先需要了解网页设计的流程，选择一款合适的开发工具以及了解常见的网页布局结构。接下来，需要掌握如何设计布局、使布局标准化的方法以及布局时的理论知识。最后通过3个实战案例，讲解 HTML 中使用 DIV 结合 CSS 实现网页常见效果的方法。

本章学习目标：

- 了解网页设计流程
- 熟悉常用的网页设计工具
- 掌握 Dreamweaver 的基本操作
- 熟悉常见网页布局结构
- 掌握居中和自适应布局的设计
- 了解网站重构的意义和方法
- 理解区块的概念及定义方法
- 掌握定位区块的方法
- 了解空白边叠加产生的原因及解决方法

4.1 网页设计流程

为了加快网站建设的速度和减少失误，应该采用一定的制作流程来策划、设计、制作和发布网站。通过使用制作流程，确定制作步骤，以确保每一步顺利完成，而不影响下一步的进行。理想的制作流程能帮助设计者解决策划网站的繁琐性，减小项目失败的风险。制作流程的第一阶段是规划项目和采集信息，接着是网站规划和设计网页，最后是上传和维护网站阶段。

每个阶段都有独特的步骤，但相连的各阶段的边界并不明显。进一步说，每一阶段并不总是有一个固定的目标。有时候，某一阶段可能会因为项目中未曾预料的改变而更改。步骤的实际数目和名称因人而异，但是总体制作流程如图 4-1 所示。

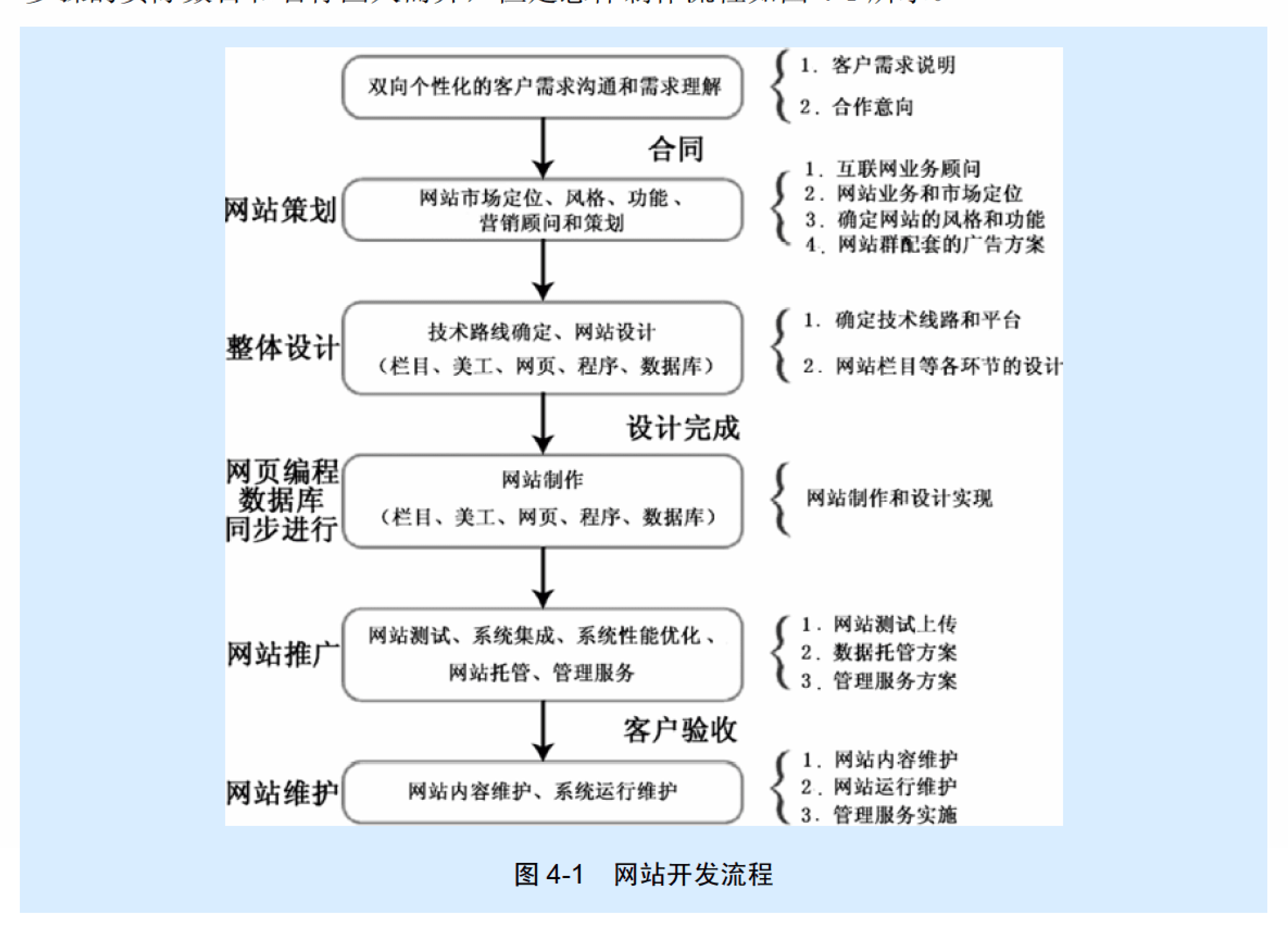


图 4-1　网站开发流程

4.2 网页设计工具

俗话说，“工欲善其事，必先利其器。”

使用不同的网页设计工具并不会影响设计网页质量的好坏。但是，如果拥有一款功能强大、使用简单的网页设计工具，往往会使我们的工作变得事半功倍，提高网页开发效率。下面介绍几款比较常用的网页设计工具。

4.2.1 记事本

使用 Windows 系统自带的记事本也可以编辑网页，如图 4-2 所示。只需要在保存文档时，以.html 或者.htm 为后缀名进行保存即可。使用记事本编辑网页，没有语法检查、没有代码格式化、没有自动提示，使网页开发非常困难，因此并不推荐使用。

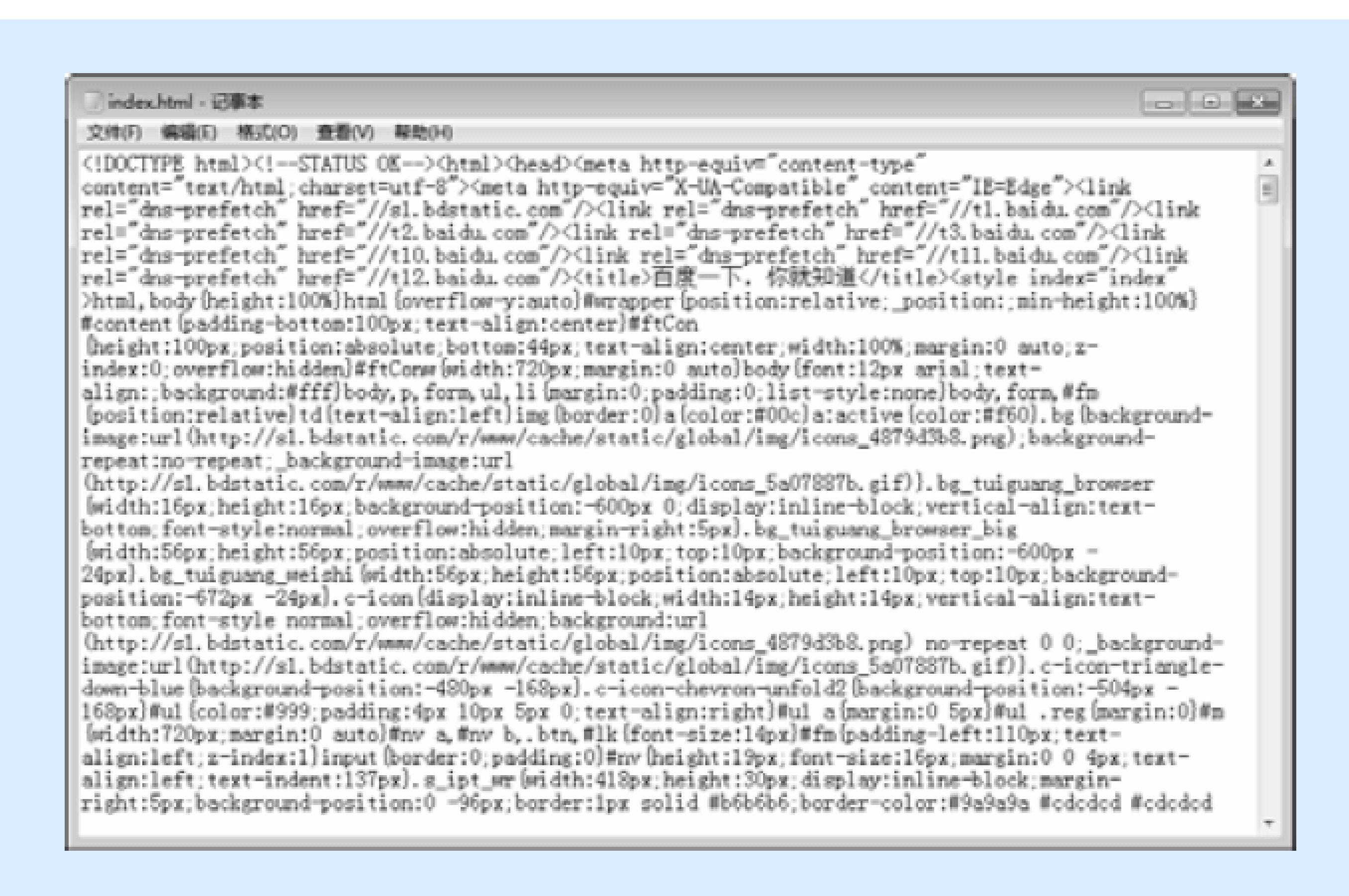


图 4-2 用记事本编辑HTML文档

4.2.2 FrontPage

FrontPage 是微软公司出品的一款网页制作入门级软件。FrontPage 作为一款中低端网页制作软件，拥有广泛的用户群。FrontPage 的特点是简单、易学、易用，正是这些特点吸引了绝大多数的网页设计初学者的眼光。FrontPage 的多个版本中，FrontPage 2003 被用户使用得最为广泛，其主界面如图 4-3 所示。

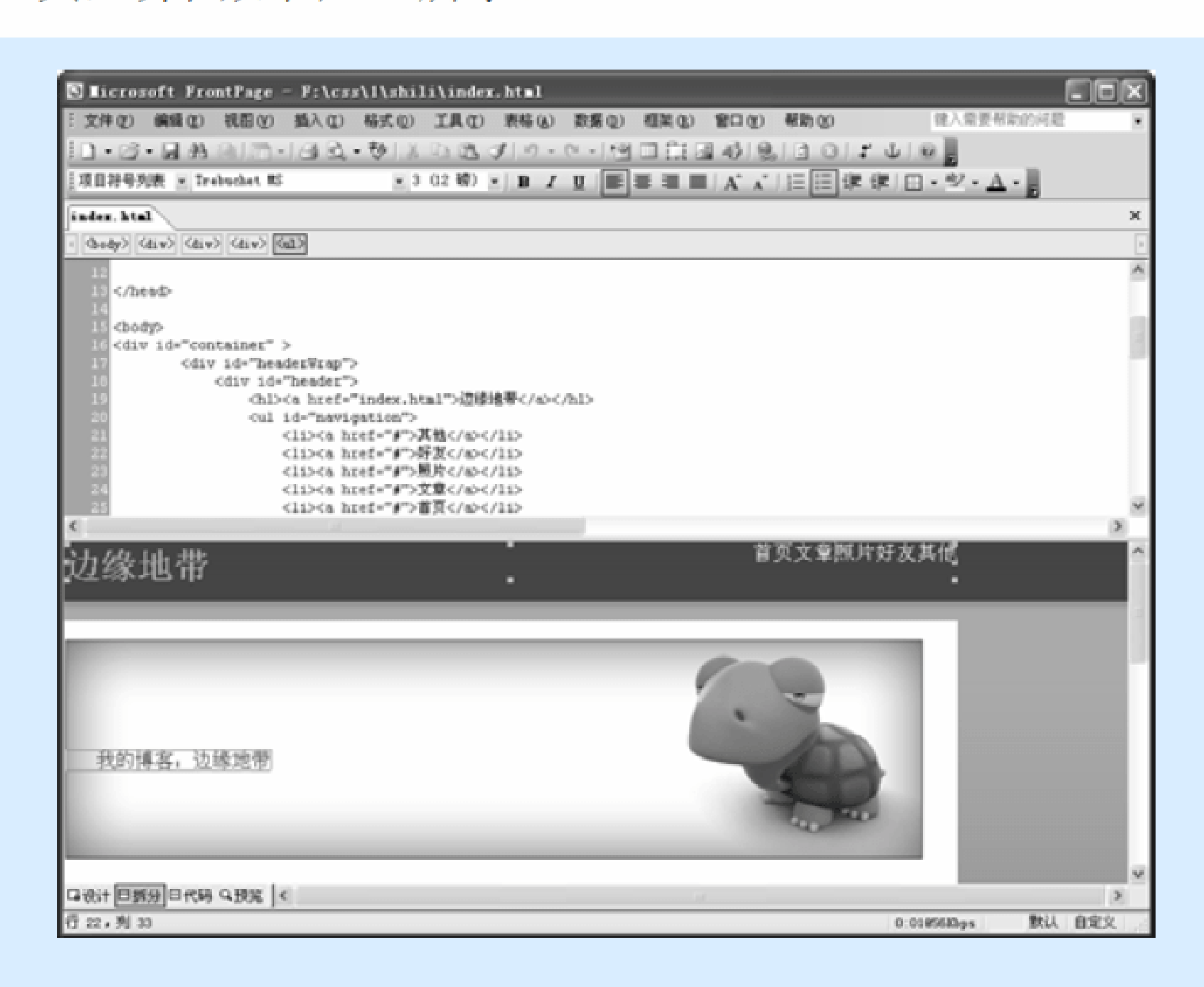


图 4-3 FrontPage 2003 的主界面

FrontPage 结合了设计、源代码和预览三种模式于一体。可以同时查看源代码和设计视图，与 Microsoft Office 各软件可无缝连接，拥有良好的表格控制能力，并且继承了 Microsoft Office 系列产品的良好易用性。

2006 年，微软在官方网站上宣布 FrontPage 停止更新。FrontPage 将被两款专业的网页设计工具所取代：Expression Web 和 SharePoint Designer。这两款软件部分组件都是基于 Microsoft FrontPage 的。在 Microsoft Office System 2007 中已经包含了 Microsoft SharePoint Designer。

4.2.3 Dreamweaver

Dreamweaver 是 Adobe 公司的产品，它与 Flash、Fireworks 并称网页三剑客，是网页制作工具界的霸主。Dreamweaver 是集网页制作和管理网站于一身的所见即所得的网页编辑器，它是第一套针对专业网页设计师特别开发的可视化网页开发工具，利用它可以轻而易举地制作出充满动感的网页。

Dreamweaver 的主界面如图 4-4 所示。

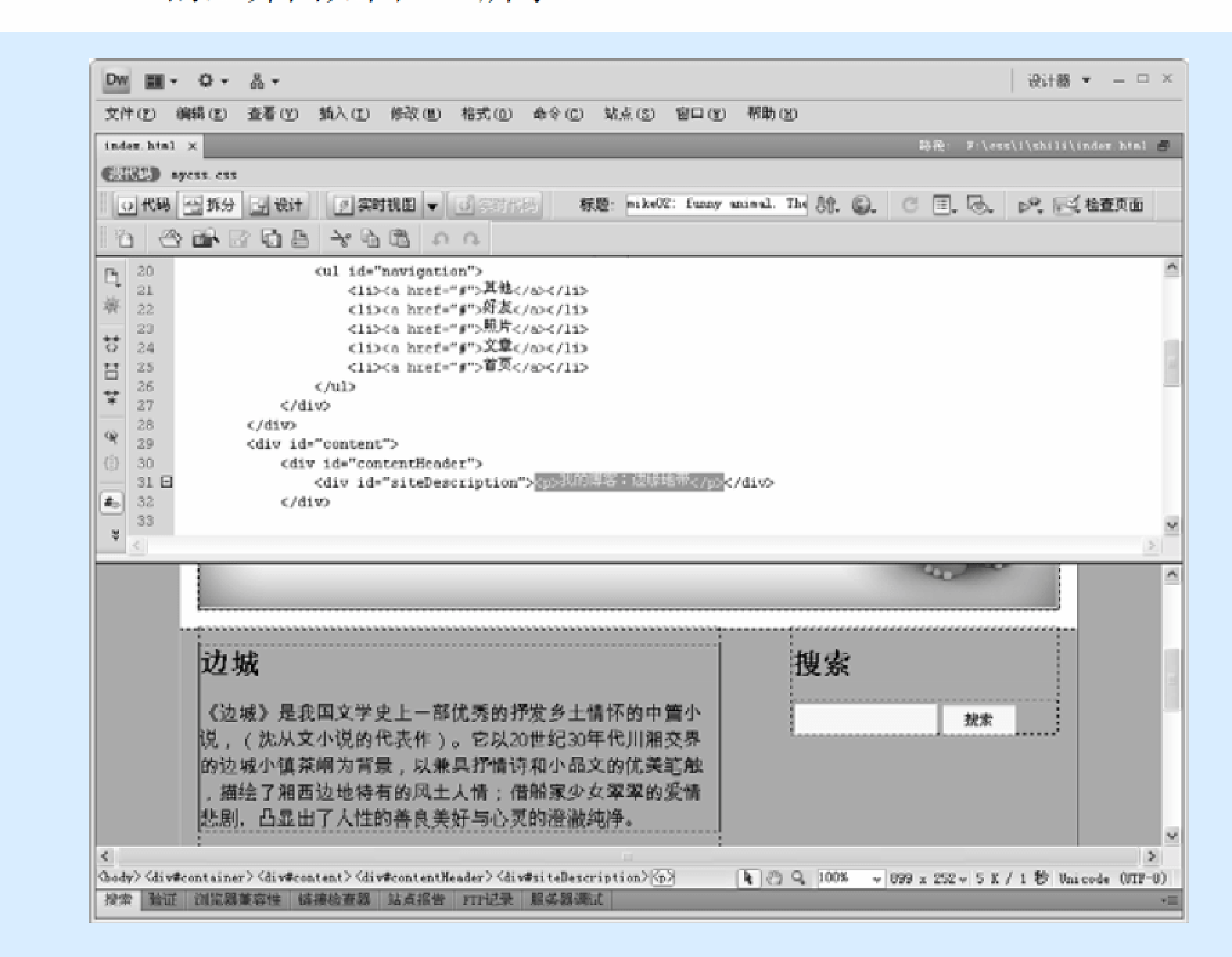


图 4-4　Dreamweaver的主界面

Dreamweaver 主要有以下优点。

- 最佳的制作效率：Dreamweaver 可以使用鼠标拖拽的方式快速地放入页面元素，它的智能提示功能使我们可以更加惬意地编写代码。
- 网站管理方便：Dreamweaver 使用网站地图可以很方便地管理站点资源和文件。
- 无可比拟的控制能力：Dreamweaver 是唯一提供 Roundtrip HTML、可视化编辑与原始码编辑同步的设计工具。

- 集成动态开发语言：Dreamweaver 还集成了动态程序开发语言，完全支持对 ASP、.NET、PHP、JS 等基本语言和连接操作数据库的功能。

总之，Dreamweaver 是一款非常优秀的网页制作工具。正如它的名字一样，它能使网页设计工作就像编织一个美丽的梦。

4.2.4 实战——制作个人主页

在了解了常用的三款网页设计工具之后，本节将以 Dreamweaver 工具为例制作一个个人主页。

在设计个人主页时，可以使用表格来控制页面的版式，再使用图文混排的方式进行布局。除此之外，还可以通过设置文本的大小、颜色、对齐方式等属性，使网页更加美观，并对一些文本可以突出显示。另外，可以使用图片和 CSS 样式来修饰效果。

具体步骤如下。

步骤 01 打开 Dreamweaver，新建一个空白的 HTML 文件，保存为 index.html。

步骤 02 在“标题”文本框中输入“欢迎光临我的主页”，按 Ctrl+S 组合键保存文档。在空白页面中右击，从弹出的快捷菜单中选择“页面属性”命令，打开“页面属性”对话框，在对话框中设置“左边距”、“右边距”、“上边距”和“下边距”都为 0，单击“确定”按钮，如图 4-5 所示。

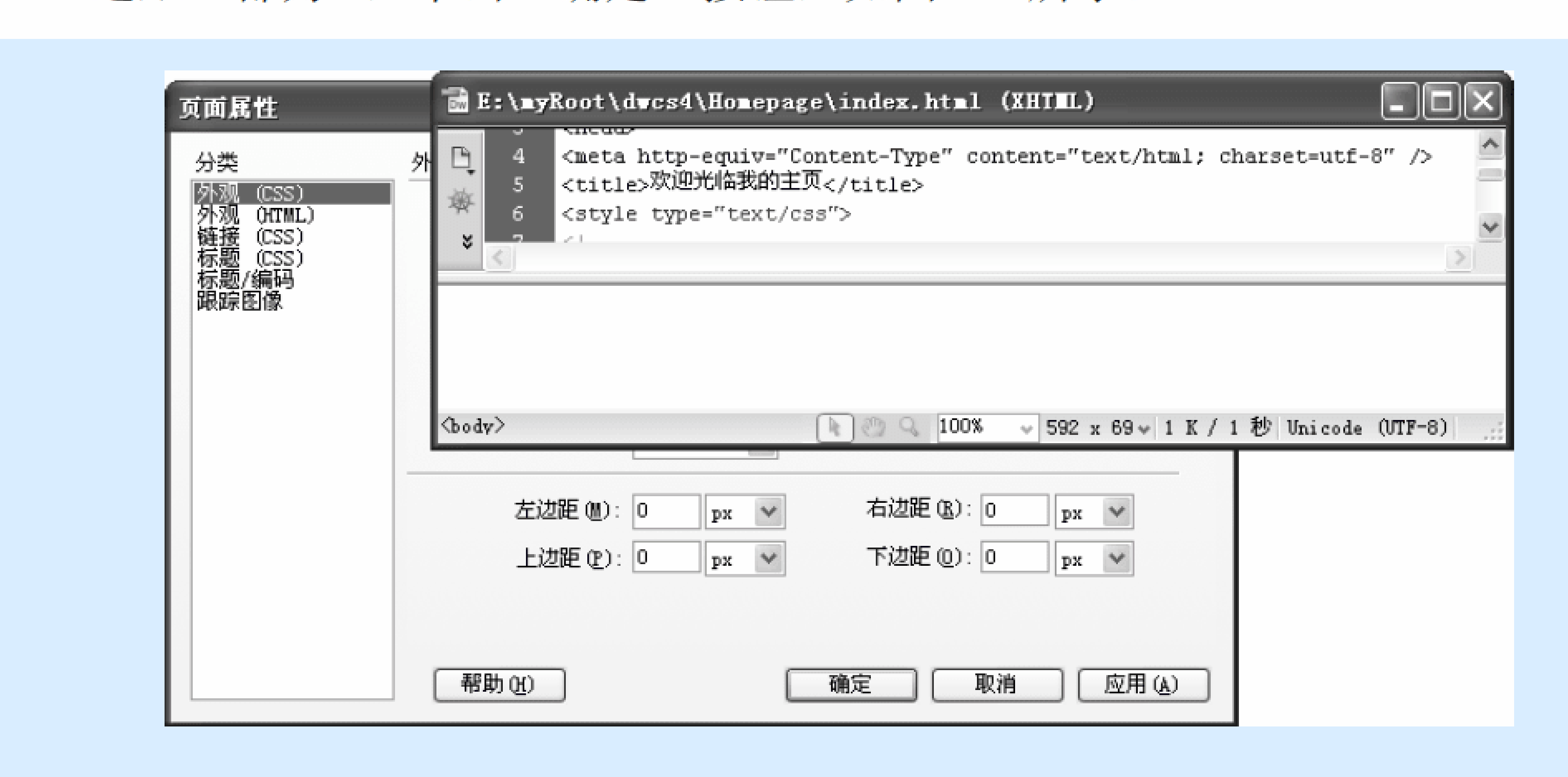


图 4-5 设置页面属性和标题

步骤 03 单击“常用”选项卡中的“表格”按钮，打开“表格”对话框，设置“行数”和“列数”均为 1，“表格宽度”为 100%，“边框粗细”为 0 像素，单击“确定”按钮创建第一个表格，如图 4-6 所示。

> 注意：利用表格可以控制文本、图像和 Flash 在页面上出现的位置，不必担心页面的整体结构遭到破坏，或者在浏览时，无法正常显示。

步骤 04 单击选中这个单元格，在“属性”面板上设置单元格的高度为 100。进入“拆分”视图，输入代码“background=“images/header.gif””为单元格设置一张背

景图片，如图 4-7 所示。

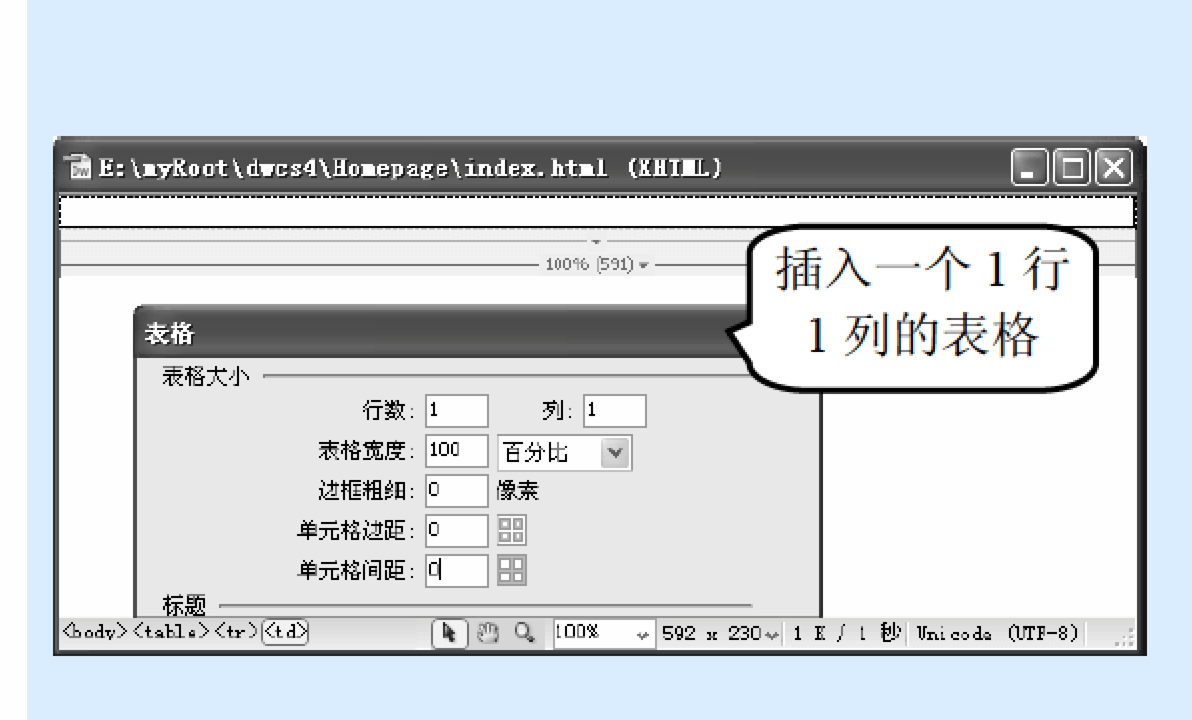


图 4-6　插入 1 行 1 列表格

图 4-7　设置表格背景图片

步骤 05　在单元格中输入“我的个人主页”，然后在“属性”面板中设置“字体”为“黑体”，“大小”为 28，“文本颜色”为#FDFDE0，如图 4-8 所示。

步骤 06　接下来制作导航栏目。单击“常用”选项卡中的“表格”按钮，创建第二个表格，设置为 6 列、间距为 1。选择表格，在“属性”面板中设置表格的“行”、“列”、“宽”、“间距”、“边框”，并把“背景颜色”设置为#006600，如图 4-9 所示。

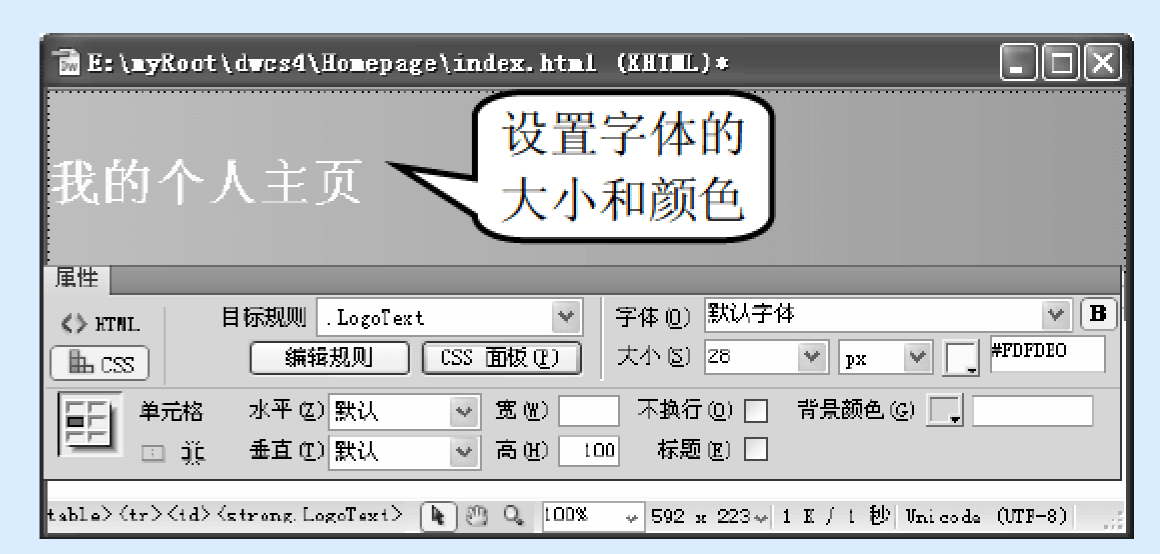


图 4-8　设置标题文字的字体属性

图 4-9　设置 1 行 6 列表格属性

注意：当用户打开“表格”对话框时，看到上面的默认设置是上一次设置的参数。每次对“表格”对话框进行更改后，由于 Dreamweaver 具有自动记忆功能，所以会记住用户这次的设置。

步骤 07　单击第二个表格，然后单击标记<tr>选中该行，在“属性”面板中，设置“高”为 18，设置“背景颜色”为#669900，设置“水平”对齐方式为“居中对齐”，设置“垂直”为“底部”，设置字体“大小”为 12、“字体颜色”为#EFF6D6，如图 4-10 所示。

步骤 08　选中前 5 个单元格，设置它们的宽度为 120，分别输入文字“我的日记”、“我的收藏”、“我的相册”、“我的朋友”和“给我留言”，如图 4-11 所示。

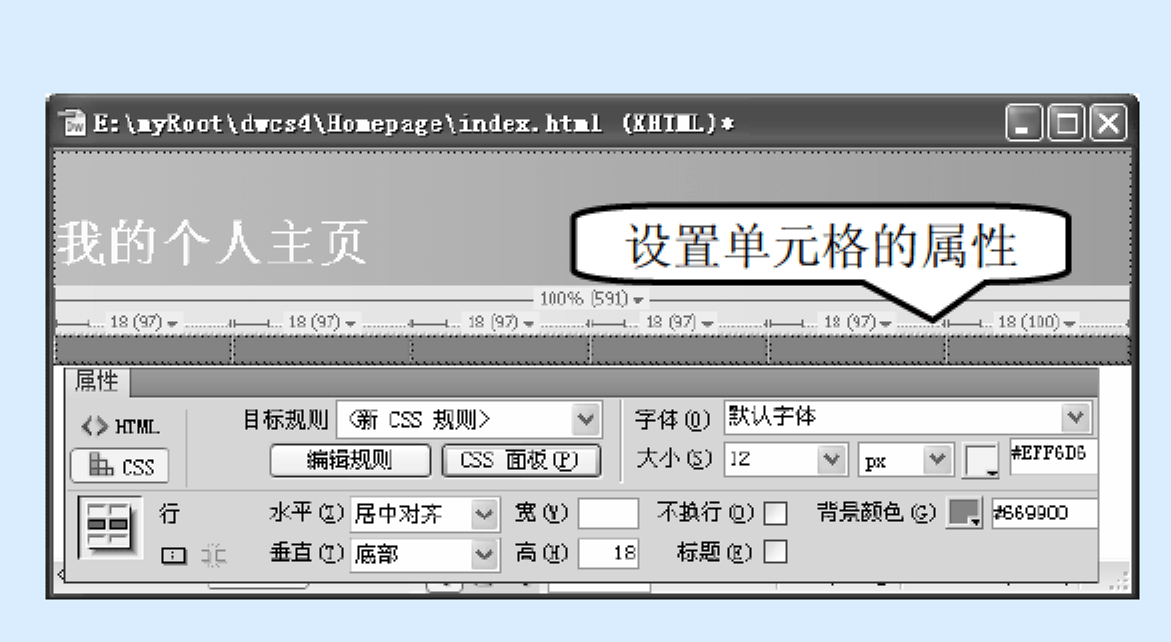


图 4-10 设置单元格属性

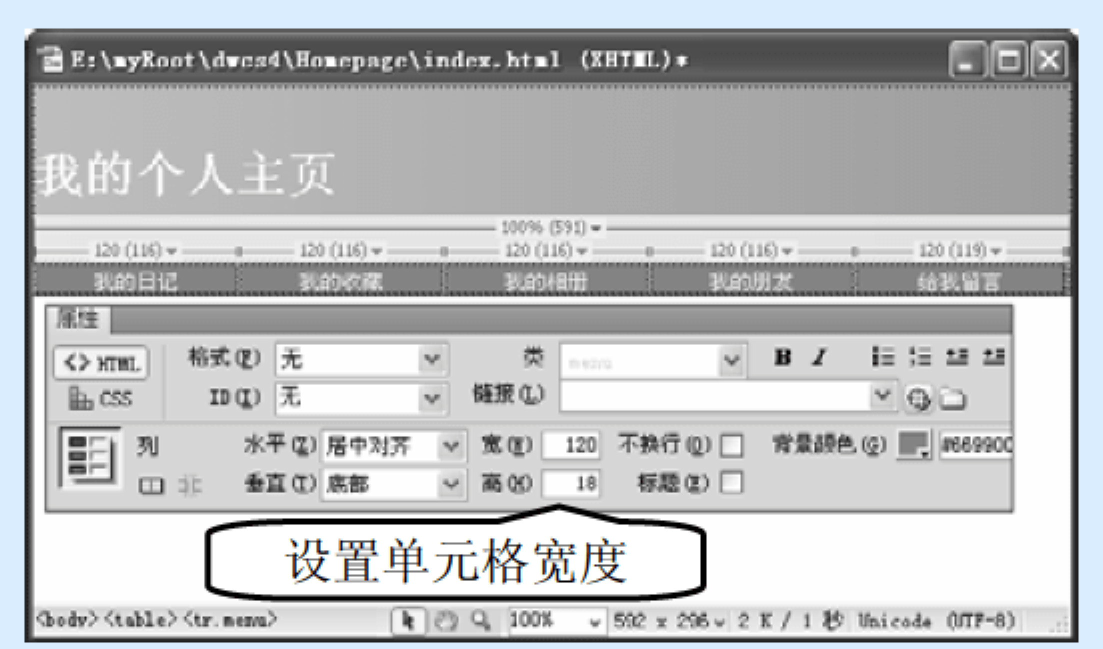


图 4-11 设置菜单文字

步骤 09 打开“页面属性”对话框，设置页面的“背景颜色”为#FDFDE0。单击“常用”选项卡中的“表格”按钮，设置参数为 1 行 2 列，设置“边框粗细”、“单元格边距”、“单元格间距”均为 0，单击“确定”按钮，创建第三个表格。然后单击右边的单元格，在“属性”面板中设置它的宽度和对齐方式，如图 4-12 所示。

步骤 10 将光标置于右边的单元格中，单击“常用”选项卡中的“图像”按钮，在打开的“选择图像源文件”对话框中，选择图像 tree.jpg，单击“确定”按钮返回，如图 4-13 所示。

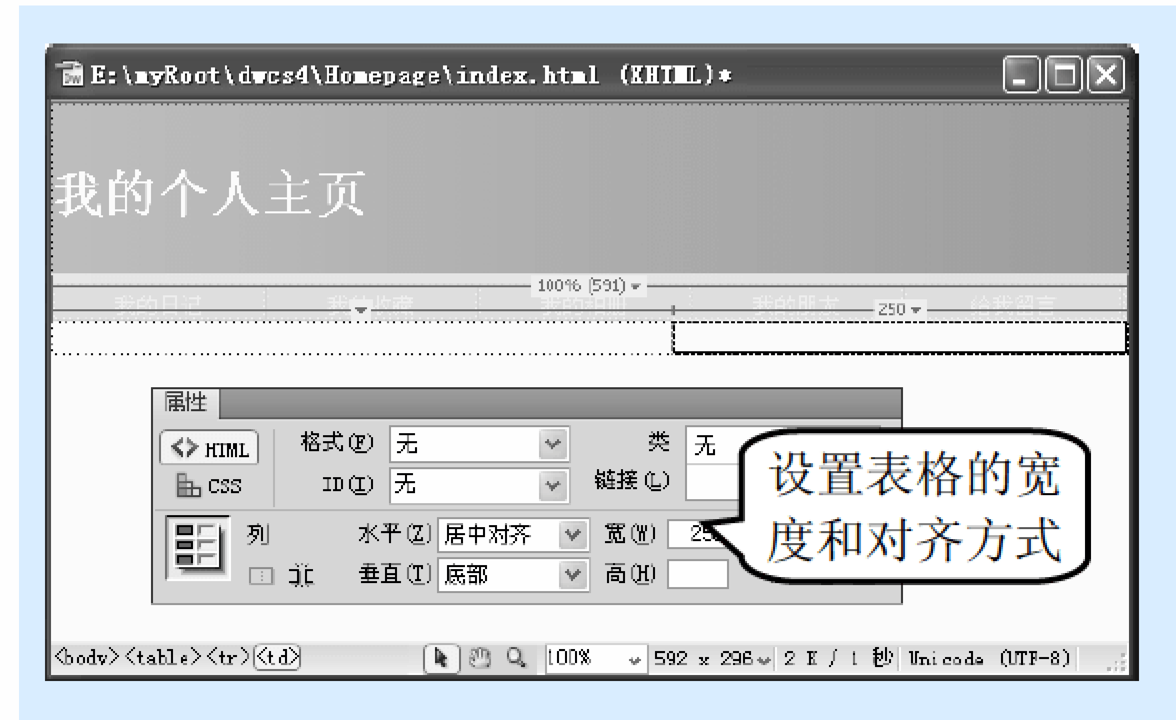


图 4-12 调整单元格宽度和对齐方式



图 4-13 插入图像

步骤 11 将光标置于左边的单元格，在“属性”面板中设置它的“垂直”对齐方式为“顶端”。单击“常用”选项卡中的“表格”按钮，设置参数为 4 行 1 列，“表格宽度”为 90%，“边框粗细”、“单元格边距”、“单元格间距”均为 0，单击“确定”按钮，创建第四个表格。在“属性”面板中设置表格的“对齐”方式为“居中对齐”，如图 4-14 所示。

步骤 12 将光标置于第一个单元格，输入文章标题，在“属性”面板中设置文字的大小、颜色、对齐方式，以及单元格的高度、对齐方式等参数，如图 4-15 所示。

步骤 13 在第二个单元格中输入文章的正文，在“属性”面板中设置文字的大小和颜色，如图 4-16 所示。

步骤 14 将光标置于第 3 个单元格中，设置它的高度为 50。单击“常用”选项卡中

的“表格”按钮，设置参数为 1 行 7 列，“边框粗细”、“单元格边距”、“单元格间距”均为 0，单击“确定”按钮，创建一个嵌套表格。在前 6 个单元格中分别输入“友情链接”、“百度”、“搜狐”、“新浪”、“计算机教程网”，按照上一步骤的方法设置字体的大小和颜色，并拖动表格边框调整到合适的宽度，如图 4-17 所示。

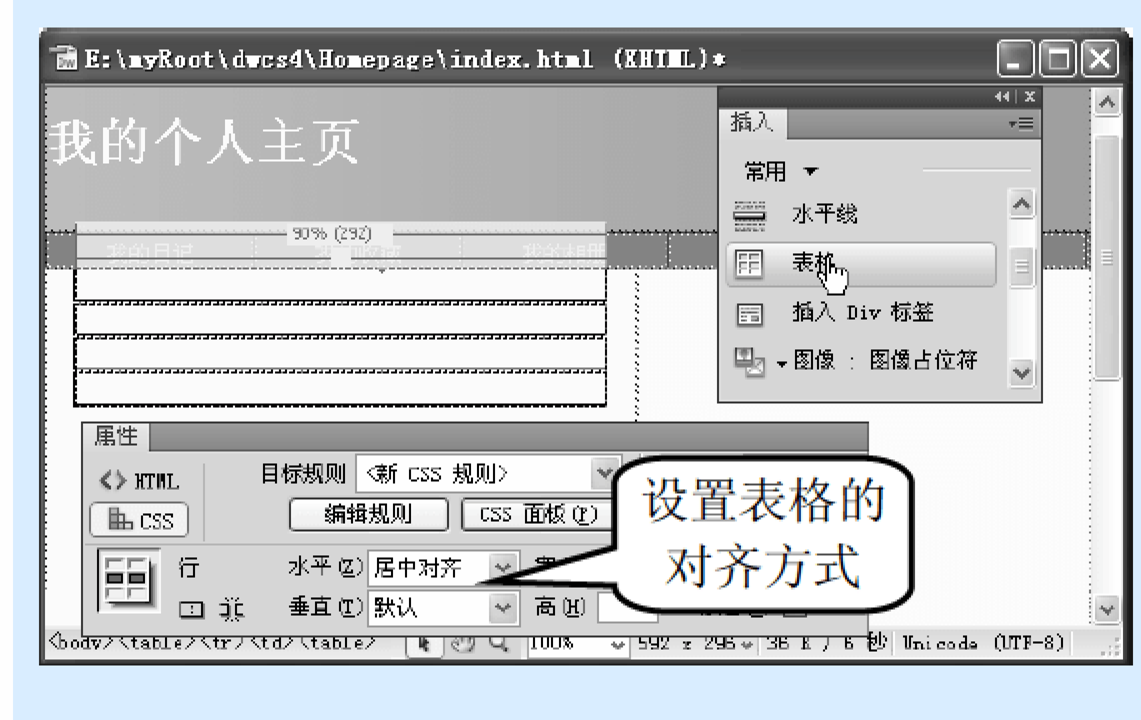


图 4-14　设置表格的对齐方式



图 4-15　设置文章标题属性

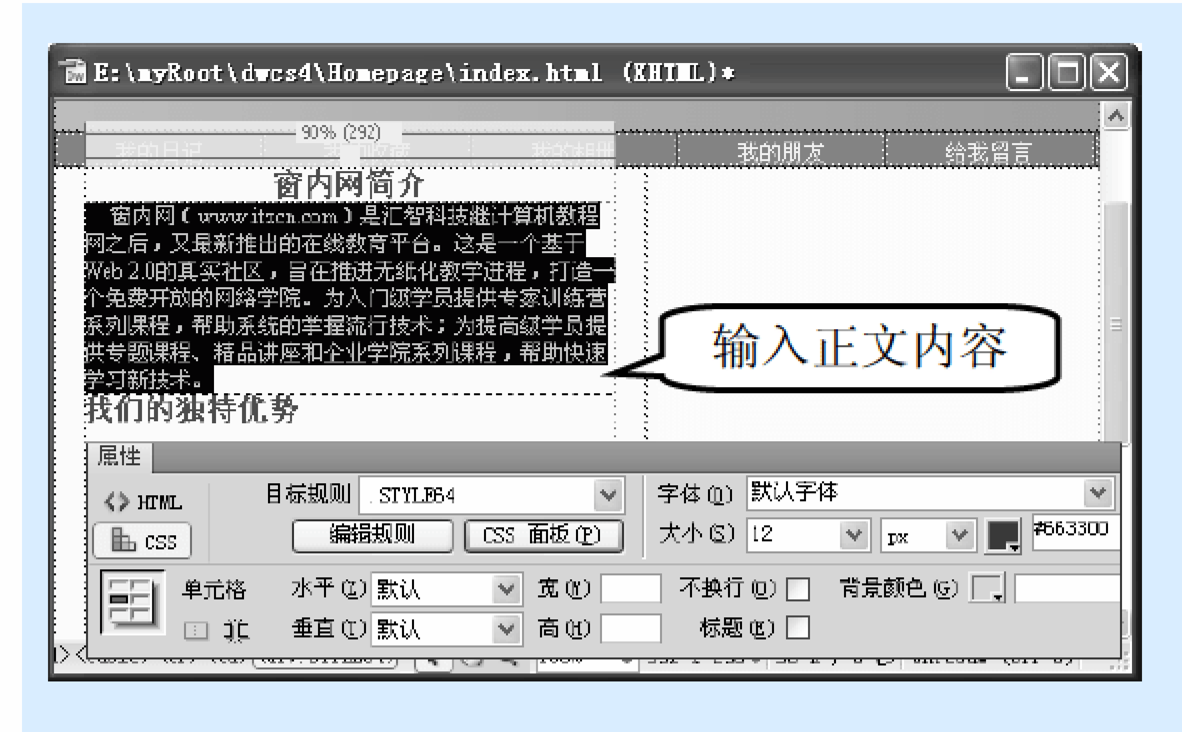


图 4-16　设置正文内容属性



图 4-17　设置友情链接表格属性

步骤 15　将光标置于第 4 个单元格中，设置它的高度为 50，输入版权信息。选中版权信息，在“属性”面板中设置“文本大小”为 12，“文本颜色”为#999999，如图 4-18 所示。



图 4-18　设置版权文字格式

步骤 16 至此，个人网站的首页制作完成，按 Ctrl+S 组合键保存文档，再按 F12 键就可以在 IE 窗口中预览网页效果了。最终效果如图 4-19 所示。



图 4-19 个人主页的效果

4.3 网 页 布 局

网页设计已经渐渐走向成熟，毫无修饰的网站必定会很快地被人遗忘，所以在制作网站之前，首先要考虑网站的整体视觉效果。在网站整体视觉效果中，主要是网页的色彩搭配与网页布局。前者决定了浏览该网站的第一印象；而后者则决定网站中的信息是否为合理的安排。

下面为读者介绍常见的网页布局结构、设计居中和自适应网页布局的方法，以及 DIV+CSS 布局网页的内容。

4.3.1 常见的网页布局结构

网页设计也属于平面设计，所以平面设计中的构成原理同样适用于网页设计。网页布局可以从两方面理解，一种是结构布局，另一种是艺术布局。

在介绍网页结构布局之前，先来了解网页布局的基本概念。页面尺寸与显示器大小及分辨率有关系，网页的局限性就在于网页无法突破显示器的范围，而且因为浏览器也将占去不少空间，留给网页的页面范围变得更小。在有限的空间内合理地安排网页元素——文本、图像、多媒体以及页眉和页脚等尤为重要，如图 4-20 所示。

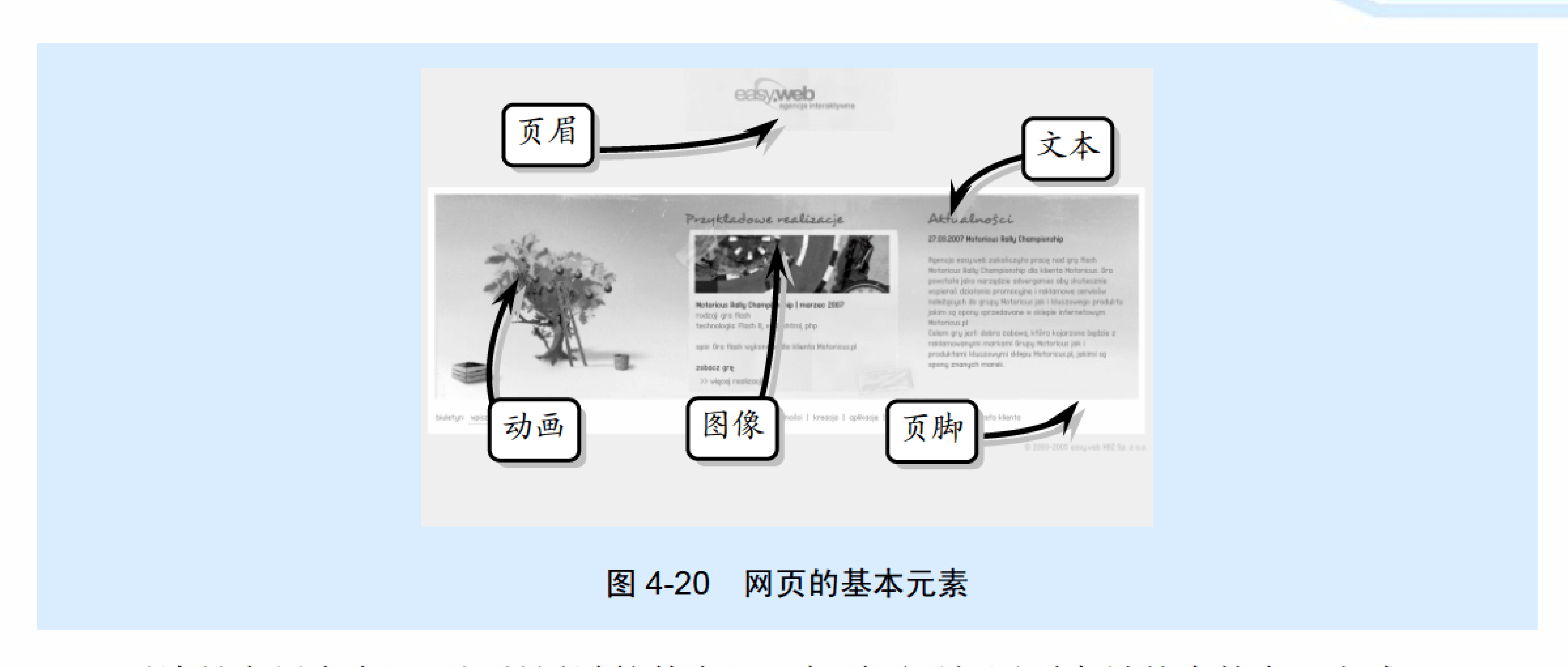


图 4-20 网页的基本元素

无论是在纸上布局，还是通过软件布局，都需要了解网页中最基本的布局方式。

1. "国"字型网页布局

"国"字型也可以称为"同"字型，是一些大型网站所喜欢的类型，即最上面是网站的标题以及横幅广告条，接下来就是网站的主要内容，左右分列两小条内容，中间是主要部分，与左右一起罗列到底，最下面是网站的一些基本信息、联系方式、版权声明等。这种结构是网上最常见的一种结构类型，如图 4-21 所示。



图 4-21 标准的"国"字型网页

2. 拐角型网页布局

拐角型结构与上一种只是形式上的区别，其实是很相近的，上面是标题及广告横幅，接下来的左侧或者右侧是一窄列链接等，正文是在很宽的区域中，下面也是一些网站的辅助信息。在这种类型中，一种很常见的类型是，最上面为标题及广告，右侧为导航链接或者广告，如图 4-22 所示。

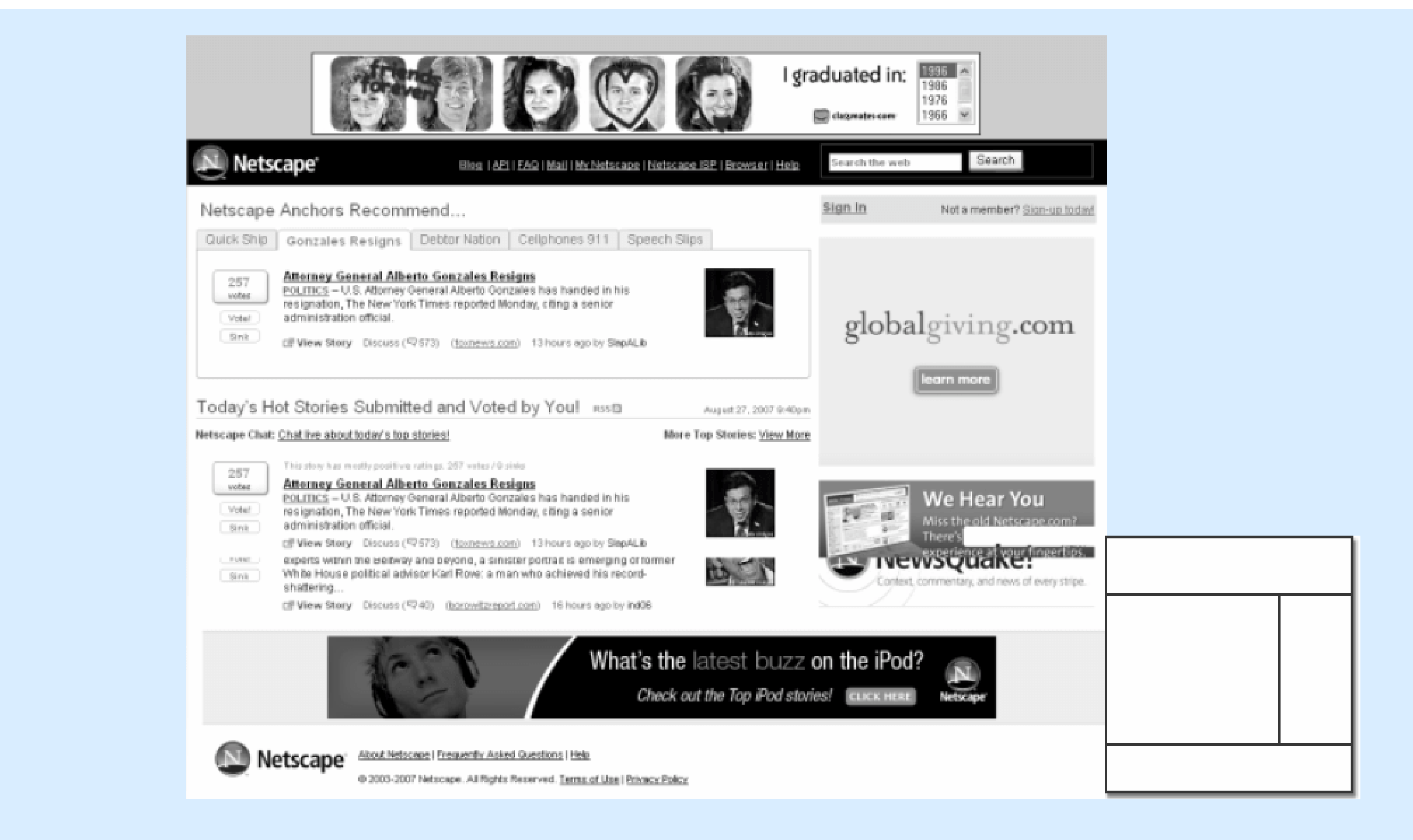


图 4-22　拐角型网页

3. 左右框架型网页布局

这是一种左右分为两页的框架结构，一般左面是导航链接，有时最上面会有一个小的标题或标志，右侧是正文。我们见到的大部分的大型论坛都是这种结构的，有一些企业网站也喜欢采用该结构，因为这种类型结构非常清晰，一目了然，如图 4-23 所示。

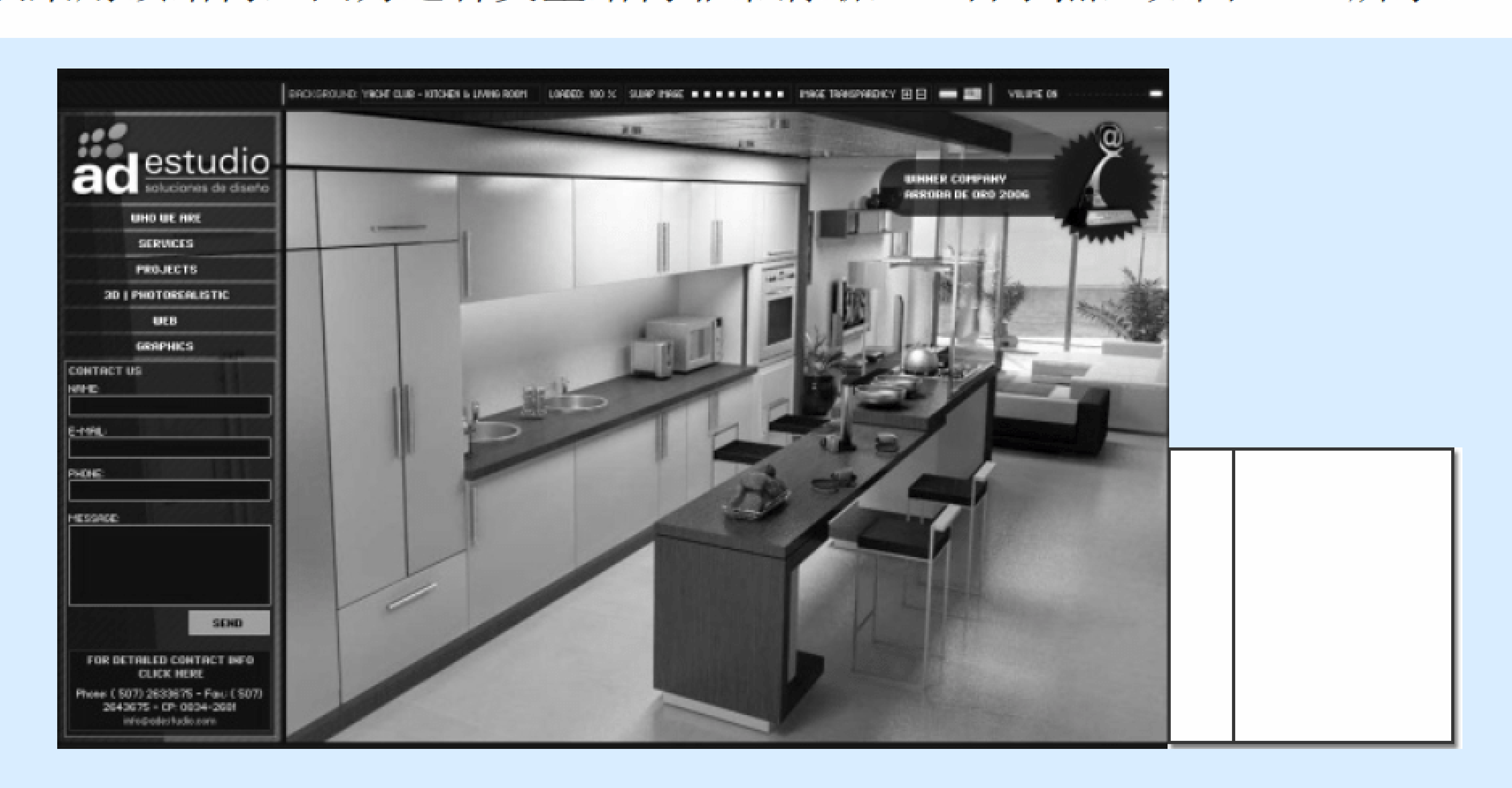


图 4-23　左右框架型网页

4. 封面型网页布局

这种类型基本上是出现在一些网站的首页，大部分为一些精美的平面设计结合一些小的动画，放上几个简单的链接或者仅是一个“进入”的链接甚至直接在首页的图片上做链接而没有任何提示。这种类型大部分出现于企业网站和个人主页，如果处理得好，会给人带来赏心悦目的感觉，如图 4-24 所示。



图 4-24　封面型网页

4.3.2　设计居中布局

布局的居中设计在目前网页布局中的应用非常广泛。所以，如何在 CSS 中设计居中的页面布局也是大多数开发人员首先要学的技能之一。

要使用 CSS 实现让页面居中，主要有两个基本方法：一个方法是使用自动空白边；另一个方法是使用定位和负值的空白边。

1. 使用自动空白边设计居中

这种方法适用于让页面上的 div 容器在屏幕上水平居中。例如，一个典型的布局可能有如下的代码：

```
<body>
<div id="Page"></div>
</body>
```

为此，只需要定义 div 的宽度，然后将水平空白边设置为自动。这样会产生如下的 CSS 样式代码：

```
#Page {
    width: 600px;
    background: #fff;
    margin: 0 auto;
}
```

这种 CSS 样式的定义方法在目前大多数浏览器中都是有效的。但是，在 IE 5 和 IE 6 中不支持自动空白边，因为 IE 将“text-align:center”理解为让所有页面元素居中，而不是文本居中。利用这一点，可以设计出兼容性更好的完美居中布局。方法是，让<body>标记中的所有内容居中，包括 div 容器，然后作为页面容器的内容重新使用左对齐。

下面展示了改进后的方案：

```
body {
    text-align: center;
}
#Page {
```

```
    width: 600px;
    background: #fff;
    margin: 0 auto;
    text-align: left;
}
```

使用上述代码可以解决多种浏览器中居中不统一的问题。其中 text-align 属性的使用是非常必要的，它不会对代码产生任何负面作用。同时，可以保证容器在页面中的绝对居中显示。

2. 使用定位和负值的空白边设计居中

与第一种方法类似，这种方法同样要定义容器的宽度。然后，将容器的 position 属性设置为 relative，将 left 属性设置为 50%，就会把容器的左边缘定位在页面的中间。如下面的 CSS 样式代码所示：

```
#Page {
    width: 600px;
    background-color: #FFF;
    position: relative;
    left: 50%;
}
```

如果不希望让容器的左边缘居中，而是希望让容器的中间对应页面居中，实现的方法是对容器的左边应用一个负值的空白边，宽度等于容器宽度的一半。这样会把容器向左移动其宽度的一半，从而让容器在屏幕上居中。代码如下：

```
#Page {
    width: 600px;
    background-color: #FFF;
    position: relative;
    left: 50%;
    margin-left: -300px;
}
```

4.3.3 设计自适应布局

顾名思义，自适应布局是指页面中容器的大小会随着浏览器的大小自动调整。自适应布局可以分为浮动布局、流体布局和弹性布局三种。

1. 浮动布局

在基于浮动的网页布局中，需要设置希望定位元素的宽度，然后再控制它们在调整时是向左或者向右进行浮动。浮动布局的经典应用分为两种情况：两列和三列的浮动布局。

2. 流体布局

在浮动布局中都是使用像素为单位来定义宽度，因此这种布局类型又称为固定布局。在使用流体布局时，尺寸是使用百分数而不是像素设置的，这样可以使流体布局能够相对于浏览器容器进行伸缩。

提示：使用流体的网页布局时，当浏览器窗口过小的时候，行会变得非常窄，很难阅读，在多列布局中尤其明显。因此，有必要适当使用以像素或 em 为单位的固定宽度，从而防止布局变得很窄。

3. 弹性布局

弹性布局是相对于字号(而不是浏览器宽度)来设置元素宽度的。通过以 em 为单位设置宽度，可以确保在字号增加时整个布局也跟着扩大。这可以将行保持在可阅读范围内，对于视力弱或有认知障碍的人尤其有用。

创建弹性布局比创建流体布局要容易得多。因为，所有 XHTML 元素基本上一直出现在相同的相对位置上，它们仅仅是同时随文本字号而增大的。将固定宽度布局转换为弹性布局也非常简单，重点是设置基字号，让 1em 大致相当于 10 像素。

与其他布局技术一样，弹性布局也有自己的问题。弹性布局的问题与固定宽度布局相同，例如，不能充分利用可用空间。另外，因为它在文本字号增加时整个布局会加大，所以弹性布局会变得比浏览器窗口宽，导致出现水平滚动条。为了防止这种情况，可能需要在 body 元素中添加 max-width 属性为“100%”。

注意：IE 6 和更低版本的浏览器当前不支持 max-width 属性，但是 Safari 和 Firefox 等符合标准的浏览器支持。

4.3.4 DIV+CSS重构网站布局

网站的重构可以理解为改变旧的 HTML 表格布局，使用新的符合 Web 标准的网站结构及代码改良的方式。将现有网站转向 Web 标准设计可以称为网站重构。网站重构的意义在于建立良好的可扩充性，通过 DIV 布局来进行数据结构的设计，便于以后更多 DIV 数据的扩充，而 CSS 对 id、class 这些元素的样式定义也使得可维护性大大提高。

与传统的表格布局页面相比，DIV+CSS 方式的最大优势就是结构清晰、维护简单和易于理解。在这种方式中，DIV(层)负责在页面中表现内容的层次和结构，CSS 则来控制内容表现方式和位置。

基于 Web 标准的网站设计的核心在于如何使用众多 Web 标准中的各项技术来达到表现与内容的分离，即网站的结构、表现、行为三者分离。只有真正实现了结构分离的网页设计，才是真正意义上符合 Web 标准的网页设计。为了达到表现与内容分离，推荐使用 XHTML 以更严谨的语言编写结构，并使用 CSS 来完成网页的布局表现，因此掌握基于 CSS 的网页布局方式，是实现 Web 标准的基础环节。

传统表格布局方式实现时是利用了 HTML 的 table 表格元素具有无边框的特性，由于 table 元素可以在显示时使得边框和间距设置为 0，即不显示边框，因此，可以将网页中的各个元素按版式划分放入表格的各个单元格中，从而实现复杂的排版组合。

table 布局的核心在于设计一个能满足版式要求的表格结构，将内容装入每个单元格

中，间距以及空格则通过插入图片进行占位来实现，最终的结构是一个复杂的表格，非常不利用于设计与修改，如图 4-25 所示。

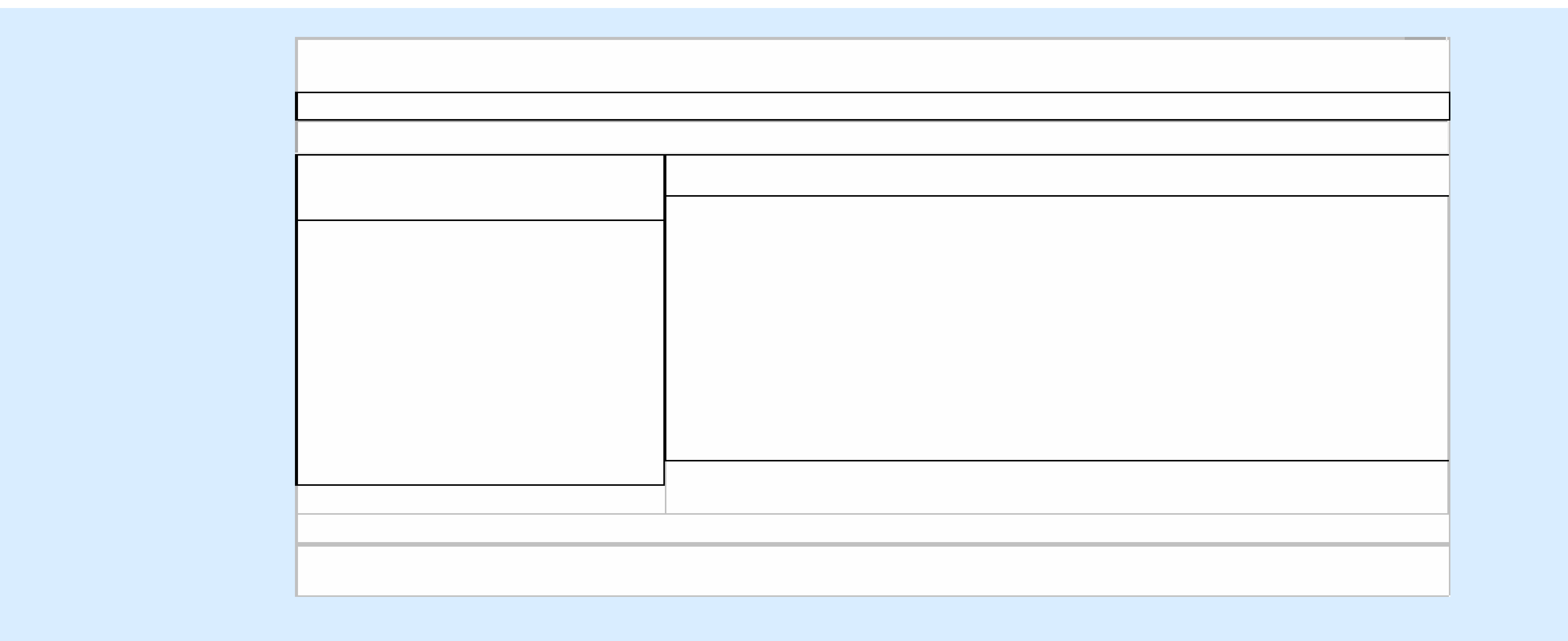


图 4-25　传统表格结构布局

另外，在表格布局的代码中最常见的是 HTML 标记之间加一些设计代码，像"width=100%"或者"border=0"等。表格布局的混合代码就是这样编写的，大量样式设计代码混杂在表格和单元格之中，使得可读性大大降低，维护起来成本也相当高。

注意：尽管有像 Dreamweaver 这样的优秀网页制作软件能够帮助设计者可视化地进行这些代码的编写，但是 Dreamweaver 永远不会智能地帮助用户缩减代码。

复杂的表格使得设计极为不易，修改更加复杂，最后生成的网页代码除了表格本身的代码，还有许多没有意义的图像占位符以及其他元素。而且文件量庞大，最终导致浏览器下载及解析速度变慢。

而使用 CSS 布局则可以从根本上改变这种情况。CSS 布局的重点不再放在 table 元素的设计中，取而代之的是 HTML 中的另一个元素 DIV。DIV 可以理解为图层或一个"区块"，是一种比表格简单的元素，从语法上是由"<div>"开始和以"</div>"结束的简单的定义。DIV 的功能仅仅是用于将一段信息给标记出来，用于后期的样式定义，这里的信息标记就是前面所提到的网页的"结构"部分。通过 DIV 的使用，可以将网页中的各种元素划分到各个 DIV 中，成为网页中的结构主体，而样式表现则由 CSS 来完成。

DIV 在使用时不再像表格一样通过其内部的单元格来组织版式，而是通过 CSS 强大的样式定义功能，可以比表格更简单、更自由地控制页面的版式及样式。

例如下面是一段 DIV 设计的代码：

```
<div id="header">内容</div>
```

上述代码在页面中定义了一个 ID 是 header 的 div 标记。配合如下代码：

```
#header {
    background-color: #DDDDDD;   /* 设置背景颜色 */
    padding: 5px;  /* 设置内间距 */
    line-height: 40px;  /* 设置行高 */
}
```

这样就定义了一个名称为“#header”的 CSS 规则，用于对页面上 ID 是“header”的对象进行样式控制。

从这个简单的 CSS 样式表代码中可以看到，HTML 中仅保留了 div 标记及其中的内容，所有的样式代码均在 CSS 样式表文件中编写。这也就是实现了 CSS 布局的“表现与内容分离”。从样式设计的角度来看，CSS 对当前 ID 名为 header 的 div 定义了许多属性，如上边距、宽度、高度以及文本对齐方式等样式。这些样式有些在 HTML 标记中可以直接实现，而类似上边距等设计则是 HTML 本身所不具备的。不仅如此，应用 CSS 布局还可以充分提高代码的利用率，效率也大大提高。

4.4 布 局 理 论

所有 CSS 布局技术都是依赖于三个基本概念：定位、浮动和空白边的操作。不同的技术其实没有本质的差异，如果理解了这些概念，则创建自己的布局和理解网页的结构是相当容易的。继上节对网页布局的介绍之后，本节将详细介绍布局时所需掌握的理论知识。

4.4.1 区块的概念

理解区块的概念是理解 DIV+CSS 制作页面的基础，区块控制页面中元素的安排和显示方式。需要注意区块的实际应用以及绝对定位和相对定位，并且掌握如何控制页面中的每个元素。通常在使用 CSS 设计页面布局时，所有的页面元素都包含在一个矩形框内，称为元素框。元素框描述了元素及其在页面布局中所占的空间大小，因此元素框可以影响其他元素的位置及大小。例如，页面中第一个元素框为 10px，那么下一个元素框就处于离顶部 10px 距离的位置。如果第一个元素框增加为 20px，则下一个元素框就要再下移 10px。而整个页面就是由这些大大小小、但不会重叠的元素框形成的。

在设计页面布局时，网页设计者要充分考虑所有页面元素的边框与元素框之间的边距的布置，使页面紧凑而又不失条理。要完全理解边距与间隙，设计者就必须清楚地明确各边间距及填充的位置，而边距、边框和填充及内容共同构成了一个区块，如图 4-26 所示。

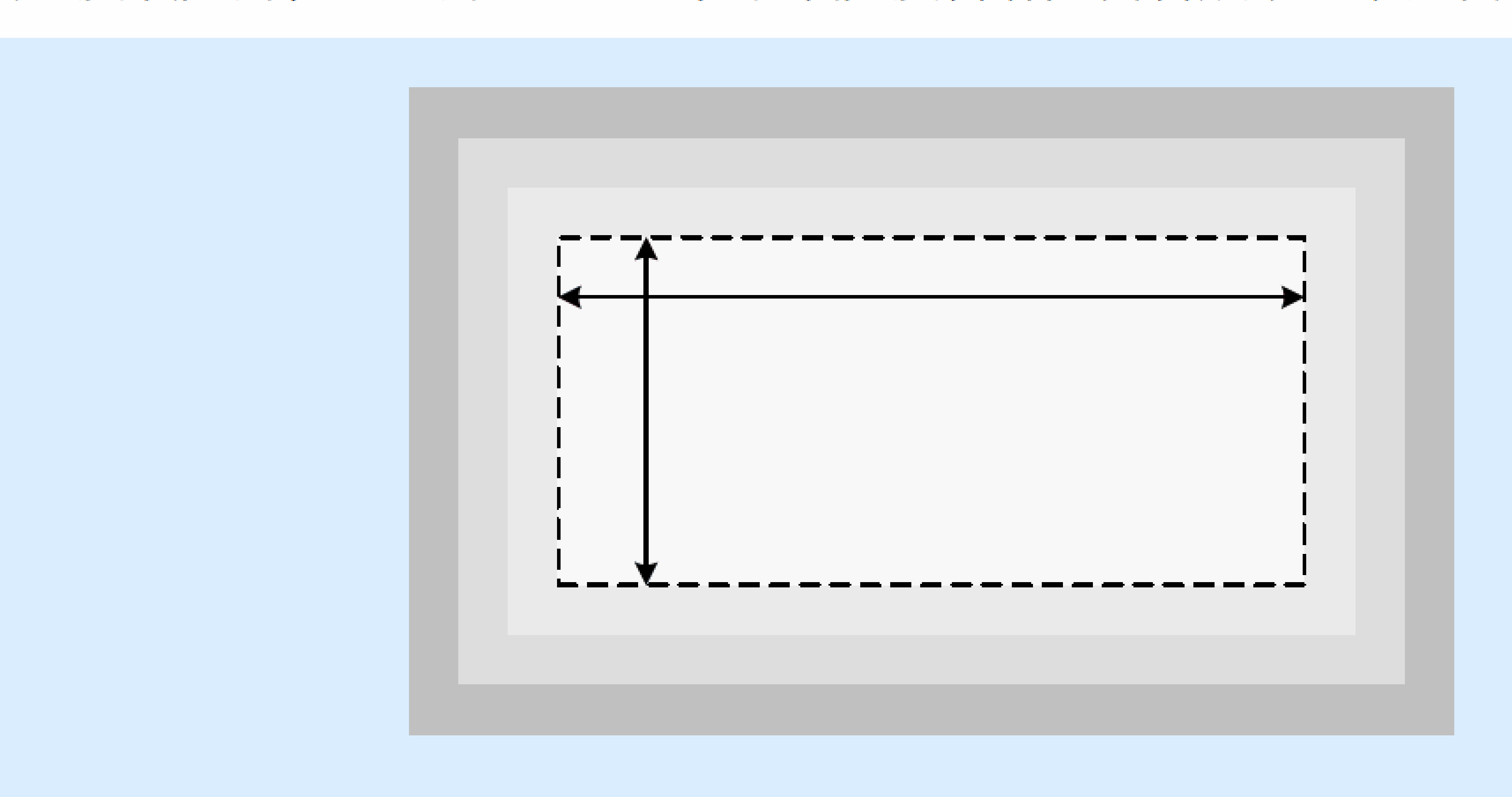


图 4-26 区块示意图

从示意图中可以清楚地看到边距与填充所控制的区域。其中元素框就是那个最深阴影的矩形框。这时，有的读者可能会问，边距和填充产生的作用都是定义距边界的距离，两者有什么区别呢？其实，边距和填充最大的区别就是背景的显示，例如，可用如下代码进行区分：

```
<p style="margin:30px; background-color:#179DF4; ">
  打造一流 IT 学习乐园  推进无纸化教学进程
</p>
<p style="margin:30px; padding:20px; background-color:#179DF4; ">
  打 造 一 流 IT 学 习 乐 园   推 进 无 纸 化 教 学 进 程
</p>
```

运行效果如图 4-27 所示。



图 4-27 边距与间隙的区别

从中可以看出，第一个段落只设定了边距，而背景色只显示在元素框区域，这说明背景色不作用于填充。第二个段落设定了填充，而背景色除了显示在元素框区域，也覆盖了填充区域，这说明背景色作用于填充。

4.4.2 定义区块

我们知道，区块由内容、边距、填充及边框共同构成。除内容外，其他都是可选的，这就会产生一个问题。例如，对于 div 标记，在 IE6 浏览器中会有 2 像素的边距，而其他浏览器则没有问题。解决方法是可以通过将元素的 margin 和 padding 设置为 0 来覆盖这个浏览器的默认值。实现时，可以分别进行，也可以使用通用选择器对所有元素进行设置。下面给出的是使用通用选择器时的 CSS 样式：

```
* {
    margin: 0;
    padding: 0;
}
```

另外，在 CSS 中的 width 和 height 属性指的是内容区域的宽度和高度。增加边框、边距和填充不会影响内容区域的大小，但是会增加区块的总大小。假设区块的每个边上有 30 像素的边距和 20 像素的填充，如果希望这个区块的宽度达到 500 像素，就需要将内容的宽度设置为 400 像素，CSS 样式如下：

```
#Page {
    width: 400px;
    margin: 30px;
    padding: 20px;
}
```

使用上述 CSS 样式后，Page 容器所定义的区块宽度即达到了 500 像素，图 4-28 中给出上述 CSS 样式的作用示意。

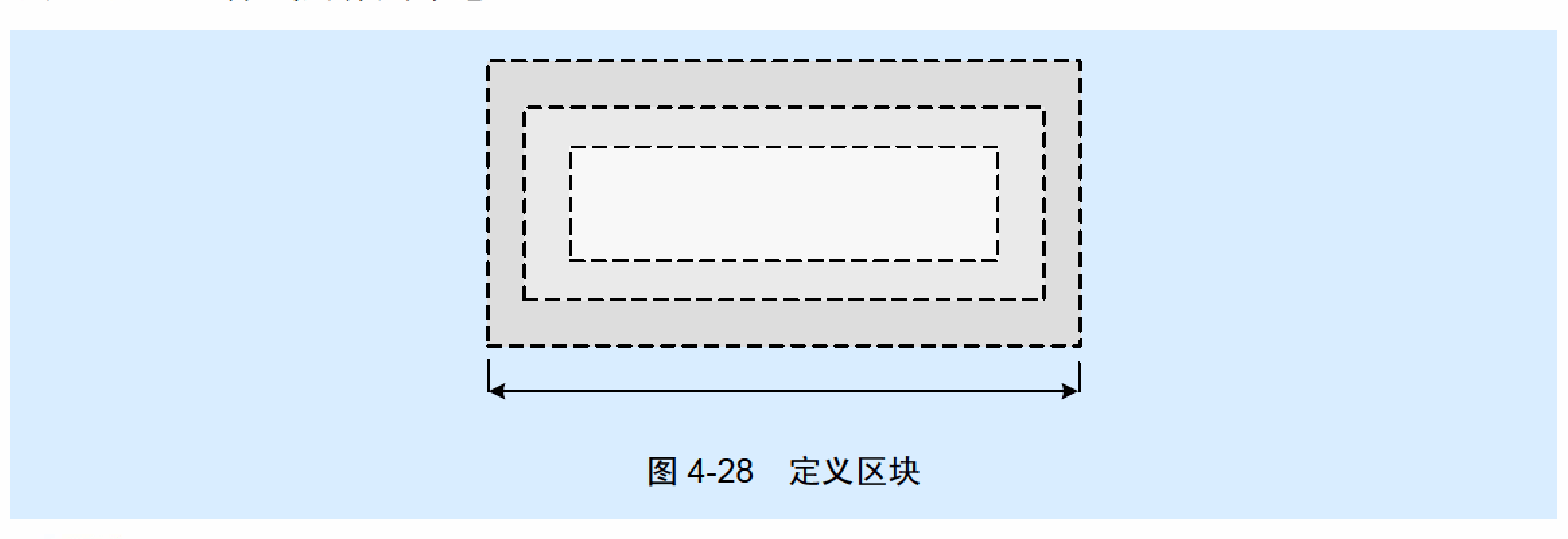


图 4-28　定义区块

4.4.3　定位

定位是 CSS 设计页面中的重要概念之一，用于控制页面元素的显示方式。如果习惯于用表格来控制布局，那么这些概念最初可能有点复杂，不易理解。可在实际使用过程中逐渐掌握区块的复杂性。定位分为绝对定位和相对定位两种类型。

1. 绝对定位

如果元素的定位类型为绝对定位，那么其显示的位置将参照浏览器左上角为开始点，其偏移方向及距离将配合边偏移属性的设定进行定位，并且是浮动于正常元素之上的。定义了绝对定位的元素将固定于页面中的某个区域，而且不会随着页面变化而变化。

如果元素中还包含有边距和间隙，则元素的定位是在元素总的区域内根据绝对定位的边偏移属性值而设定的，如图 4-29 所示。图中的灰色区域显示了包含边距和间隙的元素块应用偏移属性在页面中的绝对定位。

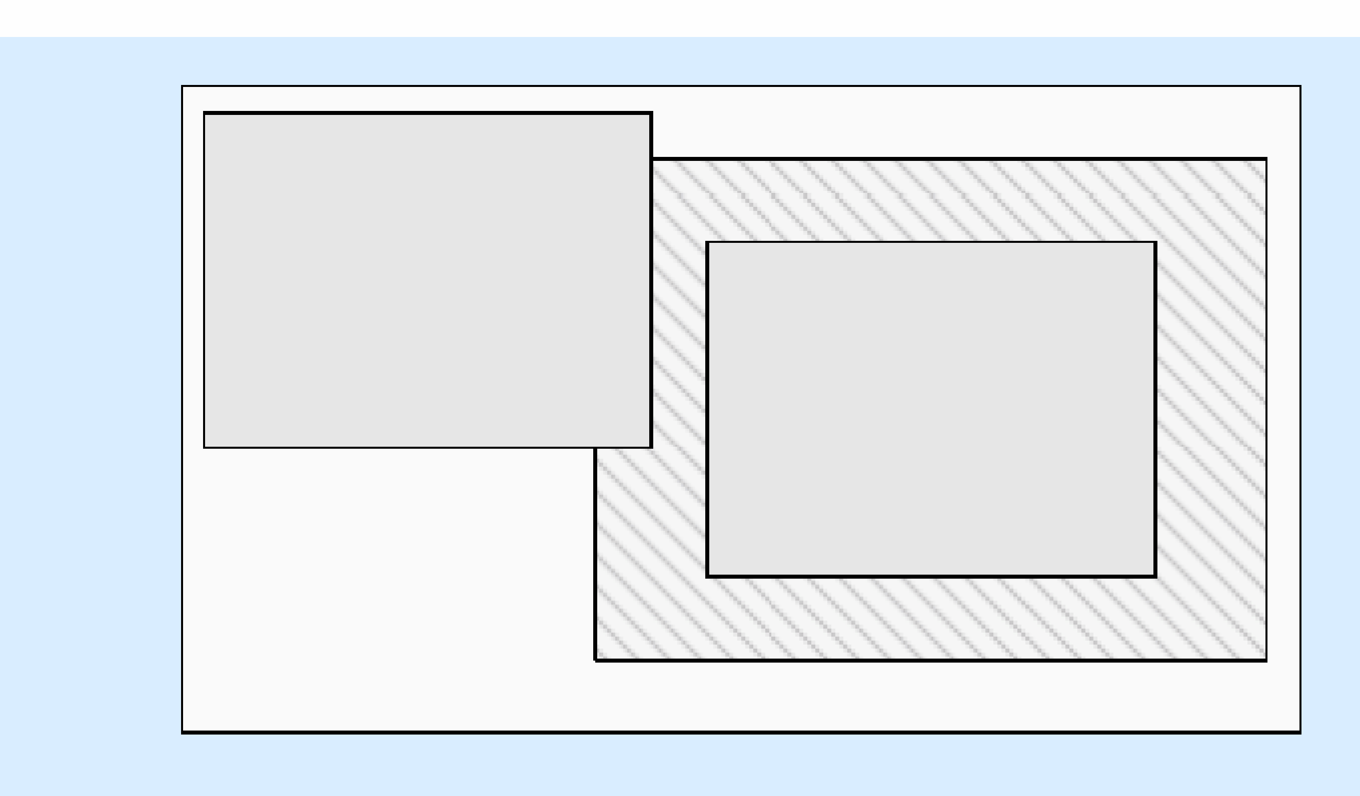


图 4-29　绝对定位示意图

注意：绝对定位元素的位置总是相对于最近的已定位父元素。如果没有已经定位的父元素，那么它的位置相对于最初的根元素，通常是 HTML 页面。

2. 相对定位

相对定位是以上级元素的原始点为参照点，然后再配合偏移属性对元素块进行定位。如果元素无上级元素，则以 body 元素为参照点；如果上级元素有边距或间隙属性，则参照点以上级元素的内容区域为原始点进行定位，如图 4-30 所示。图中显示设定了边偏移属性的元素块在应用相对定位之后是相对其上级元素的内容进行定位的。

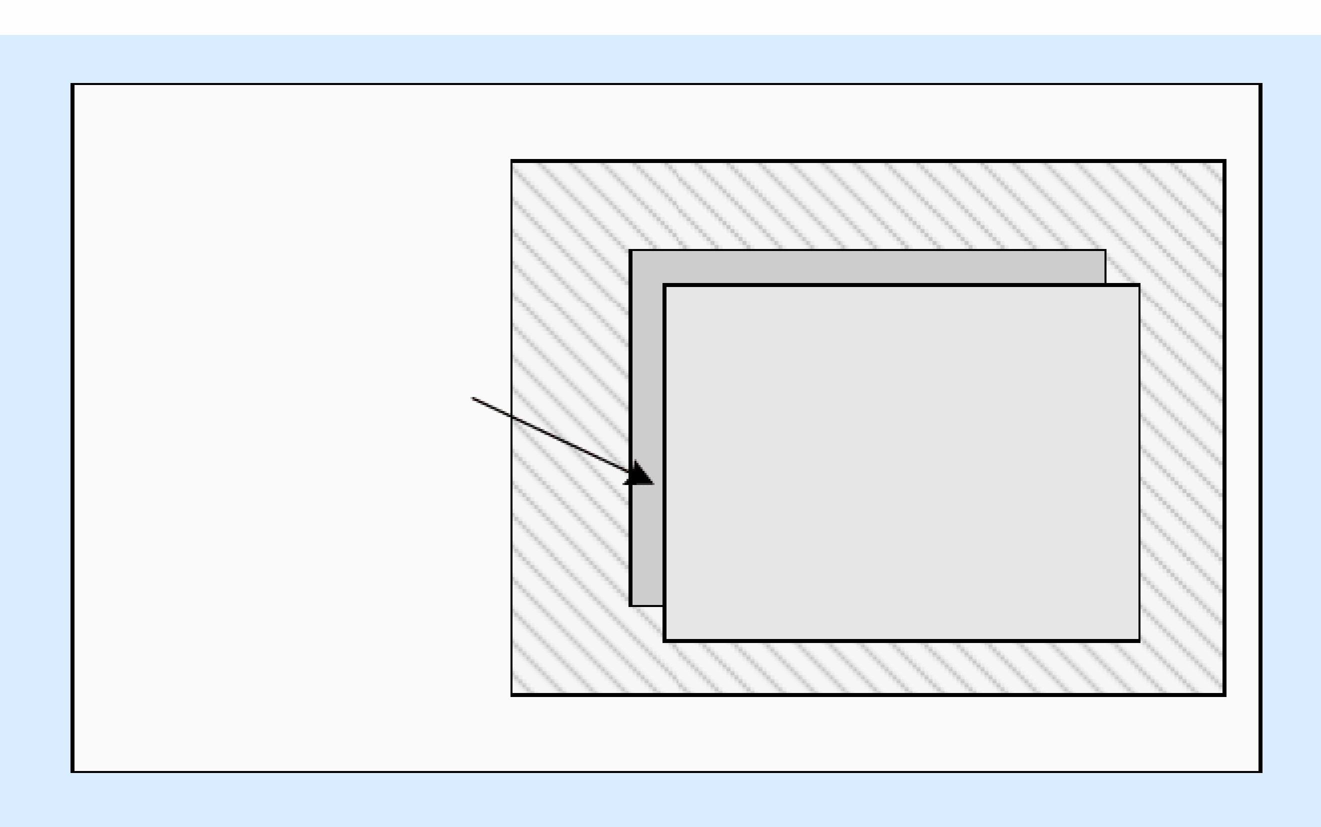


图 4-30 相对定位示意图

注意：在使用相对定位时，无论是否进行移动，元素仍然会占据原来的空间。因此，移动元素时会导致覆盖其他块。

4.4.4 实战——具有固定位置的广告框

很多网站通常会在页面中放置一些广告框。当然，如果将广告作为页面模块，显得不够醒目。所以很多站长喜欢把广告做成一直悬停在页面的侧边栏上。这样可以弥补页面的空白，也可以使广告更加醒目。

要实现上面的效果，通常的做法是使用 JavaScript 控制广告模块的位置，这种做法有一种拖拽的延迟效果。其实利用布局中的绝对定位也可以使广告框固定在浏览器窗口的某一位置。这样在浏览器中页面滚动的时候广告似乎和窗口是浑然一体的，完全不会晃动。具体步骤如下。

Position:relative;
没有设定边偏移属性

步骤 01 创建一个名为 position.html 的 HTML 文件。

步骤 02 在文件所在目录下新建 imgs 目录存放网页的图片素材，新建 css 目录存放样式表文件。

步骤 03 在 css 目录中创建一个 CSS 样式表文件，命名为“position.css”。

步骤 04 修改 position.css 文件，添加样式表代码，如下所示：

```
@charset "utf-8";
/* CSS Document */

#ad {
    background-image:url(../imgs/ad bg.gif);
    height: 126px;
    width: 211px;
    bottom: 0;
    right: 0;
    position: fixed;
    padding: 9px;
    font-family: "宋体", Arial;
}

#ad span {
    margin: 0 0 0 15px;
    font-size: 14px;
    font-weight: 700;
    color: #FFF;
}

#ad ul li {
    color: #FFF;
    font-size: 12px;
    padding: 5px;
}
```

步骤 05 打开 position.html 文件，在 head 部分引入刚才创建的 CSS 样式表文件，代码如下：

```
<link rel="stylesheet" type="text/css" href="css/position.css"/>
```

步骤 06 在页面 body 部分添加显示 QQ 群号信息的页面元素，代码如下：

```
<div id="ad">

  <span>窗内 QQ 群号</span>

  <ul>
    <li>QQ 群 1: 33925615</li>
    <li>QQ 群 2: 45368980</li>
    <li>QQ 群 3: 107423140</li>
    <li>QQ 群 4: 7858178</li>
  </ul>

</div>
```

步骤 07 在 position.html 文件中添加其他的页面内容，最后保存文件。

步骤 08 在 Chrome 浏览器中运行 position.html 页面，运行结果如图 4-31 所示。

在本示例中使用 CSS 设置广告框 ad 的背景、宽度、高度、字体、内填充，以及设置广告框的定位方式为固定定位，而且广告框右边和下边的位置都为 0，即紧贴浏览器内容框的边缘，这样该广告框将始终固定显示在浏览器的右下角。其他样式表则定义了广告框里的文本样式和边距等信息。



图 4-31　固定定位广告框的运行结果

4.4.5　空白边叠加

简单地说，当两个垂直空白边相遇时，它们将形成一个空白边。这个空白边的高度等于两个发生叠加的空白边中的较大者。

当一个元素出现在另一个元素上面时，第一个元素的低空白边与第二个元素的顶空白边发生叠加。例如下面的 CSS 样式代码：

```
#Box1 {
    height: 40px;
    border: 1px solid #000;
    margin: 30px;
    background-color: #CCC;
}
#Box2 {
    margin: 30px;
    border: 1px solid #000;
    background-color: #CCC;
    height: 40px;
}
```

两个元素的空白边叠加效果如图 4-32 所示。

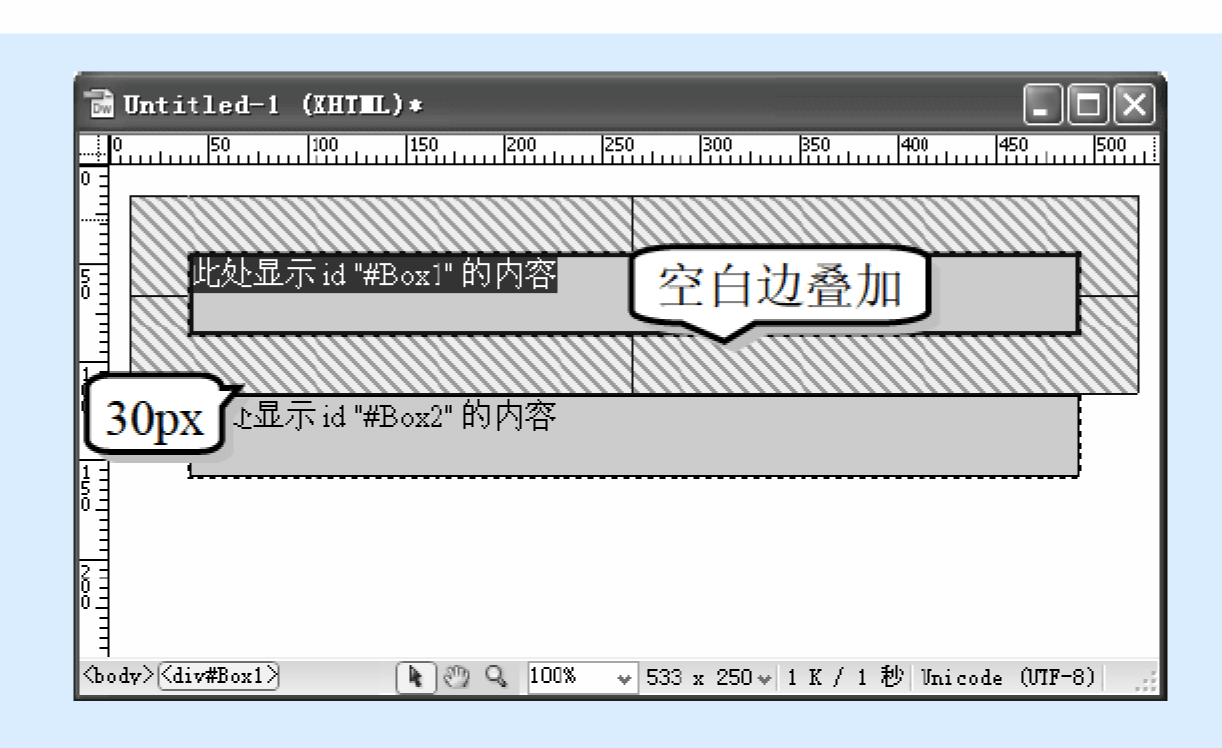


图 4-32　空白边叠加

当一个元素包含在另一个元素中时(假设没有填充或边框将空白边分隔开)，它们的顶和底空白边也会发生叠加。还有一种情况，假设有一个空元素，它有空白边，但是没有边框和填充。在这种情况下，顶空白边与底空白边就碰到了一起，此时也会发生叠加。

空白边叠加初看上去可能有点奇怪，但是它实际上是有意义的。以由多个段落组成的典型文本页面为例：第一个段落上面的空间等于段落的顶空白边；如果没有空白边叠加，那么后续所有段落之间的空白边将是相邻顶空白边和底空白边的和，这意味着段落之间的空间是页面顶部的两倍；如果发生空白边叠加，段落之间的顶空白边和底空白边就叠加在一起，这样各种距离就一致了，如图 4-33 所示。

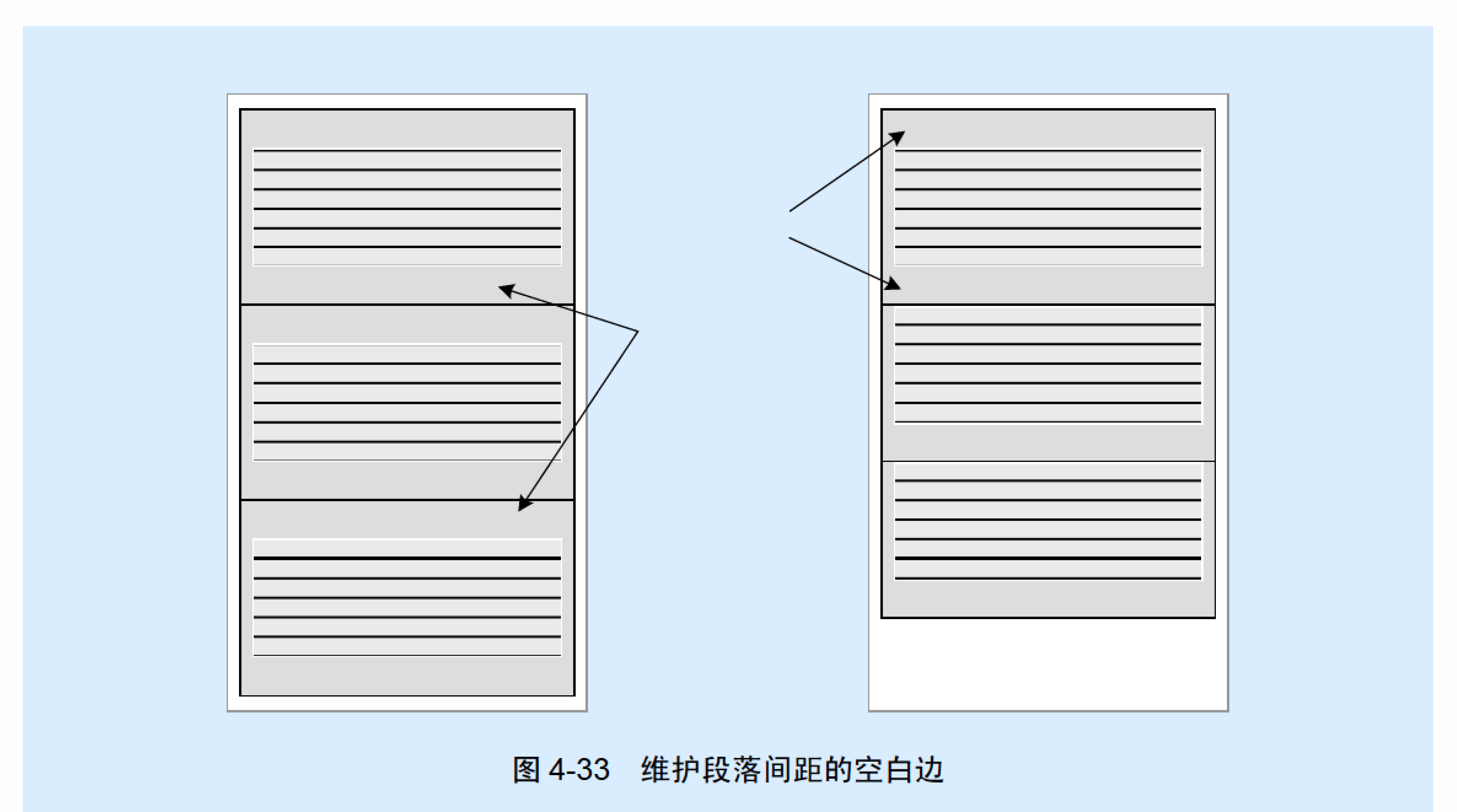


图 4-33　维护段落间距的空白边

注意：只有普通文档流中区块的垂直空白边才会发生叠加。而行内块、浮动块或绝对定位块之间的空白边是不会叠加的。

解决空白边叠加问题的方法是在叠加段落间添加如下代码即可：

```
<div style="clear:both"></div>
```

但是要注意，有时候的空白边叠加问题不需要解决，例如在显示大段文本或段落时。因此，读者在实际设计页面时，应该区分何时希望产生叠加，并加以处理。

4.5　实战——制作网页导航条

每个网页都有一个导航条，它通常位于页面 Logo 的下方，呈水平方向显示。导航条可以使网站的结构比较清晰，更容易阅读和维护。

本次实战将使用 HTML 中的 ul 列表元素配合 CSS 实现一个漂亮的网页导航条。

具体步骤如下。

步骤 01 创建一个名为 menu.html 的 HTML 文件。

步骤 02 打开 menu.html 文件并进行编辑，添加如下导航条代码：

```
<div id="menubar">
  <ul>
    <li><a href="#"><span>Home</span></a></li>
    <li><a href="#"><span>MyPhoto</span></a></li>
    <li><a href="#"><span>MyArticle</span></a></li>
    <li><a href="#"><span>MyFamily</span></a></li>
    <li><a href="#"><span>MyScroll</span></a></li>
    <li><a href="#"><span>MyFriend</span></a></li>
    <li><a href="#"><span>Message</span></a></li>
  </ul>
</div>
```

步骤 03 在 HTML 文件下创建 css 目录，并在 css 目录中创建一个名为 style.css 的文件。

步骤 04 编辑 style.css 文件，添加如下样式代码：

```
@charset "utf-8";

body {
    background-image: url(../imgs/display bg.jpg)
}
ul, li, a {
    margin: 0;
    padding: 0;
}
#menubar {
    margin: 180px 0 0 0;
    height: 24px;
    font-size: 14px;
    border-bottom: 1px solid #E276A7;
}
#menubar li {
    display: inline;
}
#menubar a {
    float: left;
    background: url("../imgs/m left.gif") no-repeat left top;
    padding: 0 0 0 4px;
    text-decoration: none;
}
#menubar a span {
    display: block;
    background: url("../imgs/m right.gif") no-repeat right top;
    padding: 5px 15px 4px 6px;
    color: #333;
}
```

步骤 05 在 menu.html 页面的 head 部分中添加对 CSS 样式表文件 style.css 的引用，代码如下：

```
<link rel="stylesheet" type="text/css" href="css/style.css"/>
```

步骤 06　在 Chrome 浏览器中运行 menu.html 页面，运行结果如图 4-34 所示。

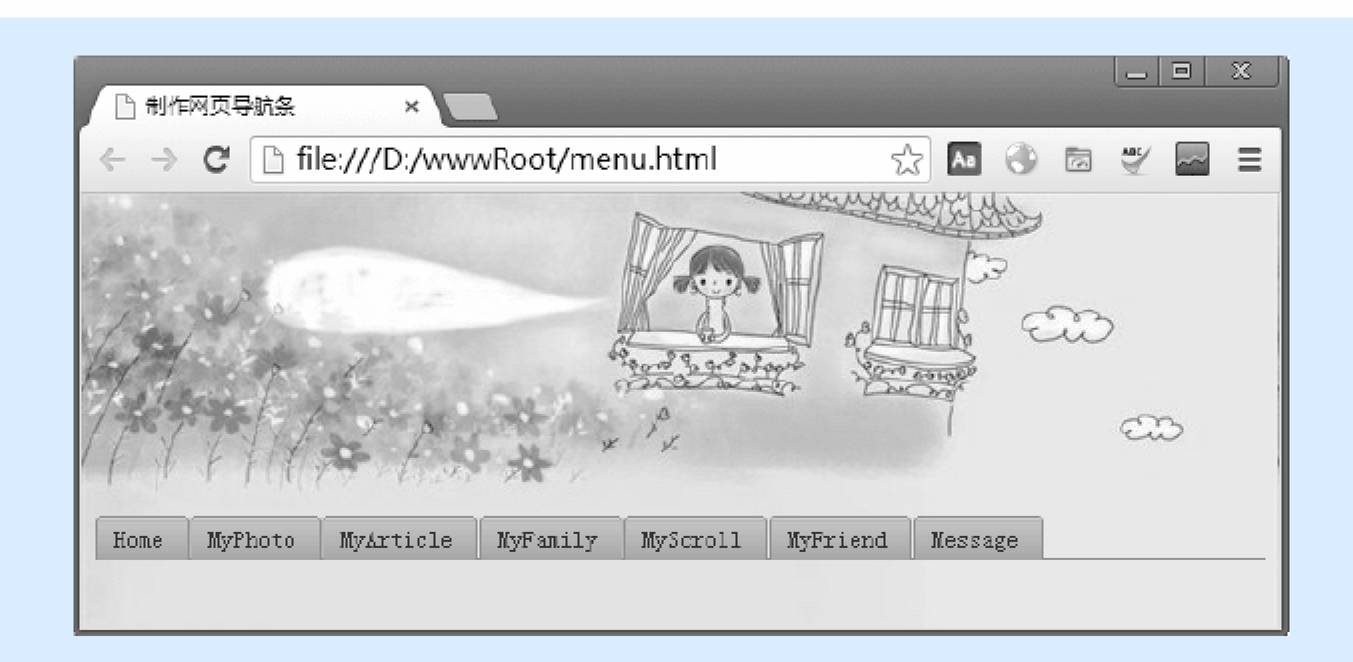


图 4-34　横排的列表元素

4.6　实战——制作文本环绕图片

像很多报纸上的版块一样，一篇新闻往往附带一张或多张图片，这些图片多是对新闻内容的一个说明。为了将适当的图片放置到适当的位置上，而且既美观又节省页面资源，往往会使用文字环绕的效果布局页面。当然，很多网站或博客也大量使用这种方式来布局网页。

环绕效果节省了页面中大量的空白，使页面显得更加紧凑。下面我们利用布局的浮动来实现一个文字环绕的图片效果。

具体步骤如下。

步骤 01　创建一个名为 index2.html 的 HTML 文件。

步骤 02　打开 index2.html 文件并进行编辑，添加如下页面代码：

```
<div id="content"> <img src="imgs/15.jpg" class="box" />
  <p>　　下载是指通过网络进行传输文件，把互联网或其他电子计算机上的信息保存到本地电脑上的一种网络活动。下载可以显式或隐式地进行，只要是获得本地电脑上所没有的信息的活动，都可以认为是下载，如在线观看视频。</p>
  <p>　　下载是指通过网络进行传输文件，把互联网或其他电子计算机上的信息保存到本地电脑上的一种网络活动。下载可以显式或隐式地进行，只要是获得本地电脑上所没有的信息的活动，都可以认为是下载，如在线观看视频。</p> <p>　　下载是指通过网络进行传输文件，把互联网或其他电子计算机上的信息保存到本地电脑上的一种网络活动。下载可以显式或隐式地进行，只要是获得本地电脑上所没有的信息的活动，都可以认为是下载，如在线观看视频。</p>
</div>
```

步骤 03　在 css 目录中创建一个 CSS 文件，命名为 box.css。

步骤 04　编辑 box.css 文件，添加如下 CSS 样式代码：

```
@charset "utf-8";

body {
    background-image: url(../imgs/float bg.gif);
    padding: 70px 0 0 50px;
```

```
}
#content {
    font-size: 13px;
    font-family: "宋体", Arial;
    color: #666;
    width: 550px;
    line-height: 21px;
}
#content .box {
    float: right;
    width: 173px;
    height: 192px;
    margin: 10px;
    border: 2px solid #999;
}
```

步骤 05 在 index2.html 页面的 head 部位引入 box.css 文件，代码如下：

```
<link rel="stylesheet" type="text/css" href="css/box.css"/>
```

步骤 06 在 Chrome 浏览器中运行 index2.html 页面，运行结果如图 4-35 所示。



图 4-35 文字环绕效果

4.7 实战——制作三栏博客页面

我们曾经在网页设计中大量地运用表格来进行布局，而学习了 DIV+CSS 布局之后，我们要思考的是如何不用表格，而用 DIV 容器以及 CSS 和 XHTML 来控制实现一个不但文件体积小，而且内容和页面更具亲和力的效果。本次实战我们来一步一步地学习如何基于 CSS 建立一个包含三列的博客页面。

为了方便讲解，示例中使用了 Dreamweaver 工具。主要步骤如下。

步骤 01 从菜单栏中选择“文件”→“新建”命令，打开“新建文档”对话框，选择“空白页”选项卡，在“页面类型”中选择“HTML”，在“布局”中选择“无”，在“文档类型”下拉列表框中选择“XHTML 1.0 Transitional”，单击“创建”按钮，如图 4-36 所示。



图 4-36 “新建文档”对话框

步骤 02 在“文档”工具栏的“标题”文本框中输入“我的个人博客”，保存文档。首先创建基本的 HTML 结构，切换到代码视图和设计视图，在“文档”窗口中<body>标记内输入如下代码(见图 4-37)。基本的布局包含 5 个 div，即标题、页脚和三栏。标题和页脚占据整个页宽。

```
<div id="header"></div>
<div id="left"></div>
<div id="right"></div>
<div id="middle"></div>
<div id="footer"></div>
```

步骤 03 在 id 为 header 的标记中输入文字，设为 1 级标题，作为网页的标题内容，如图 4-38 所示。



图 4-37 定义页面标题和结构

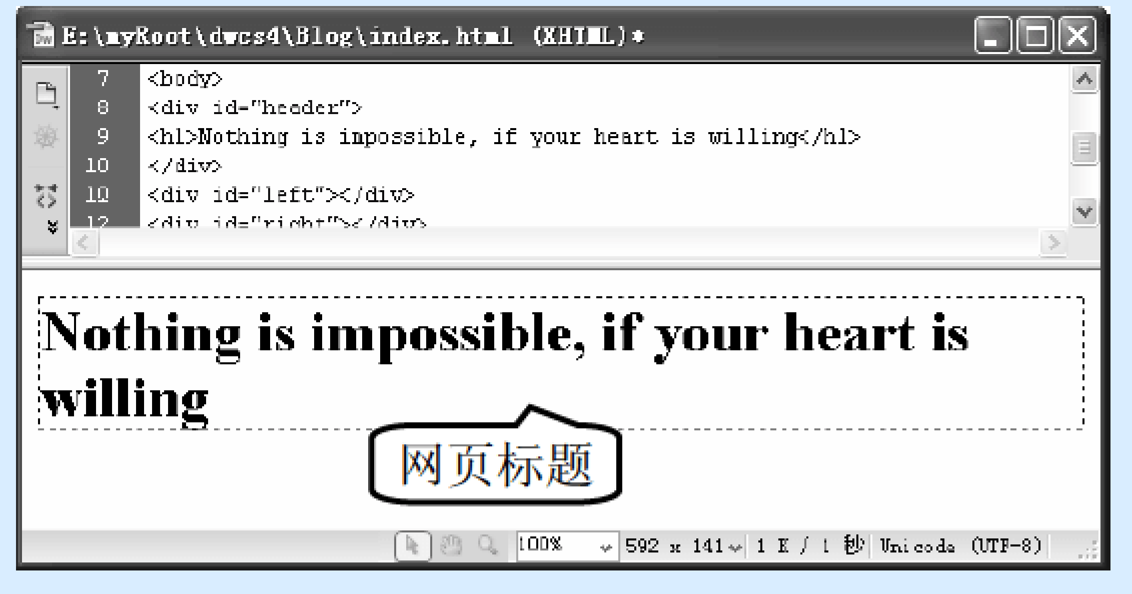


图 4-38 设置博客标题文字

步骤 04 在 id 为 left 的标记中输入“网站目录”，设为 3 级标题，然后插入无序列表，输入网站目录作为网页的左侧边栏，代码如下(见图 4-39)：

```
<h3>网站目录</h3>
<ul>
<li><a href="#">日志</url></a></li>
```

```
<li><a href="#">相册</url></a></li>
<li><a href="#">收藏</url></a></li>
<li><a href="#">好友</url></a></li>
<li><a href="#">群组</url></a></li>
<li><a href="#">我参与</url></a></li>
<li><a href="#">个人资料</url></a></li>
</ul>
```

步骤 05　在 id 为 right 的标记中输入“最近发表”，设为 3 级标题，然后插入无序列表，输入最新发表的文章列表，作为网页的右侧边栏，如图 4-40 所示。

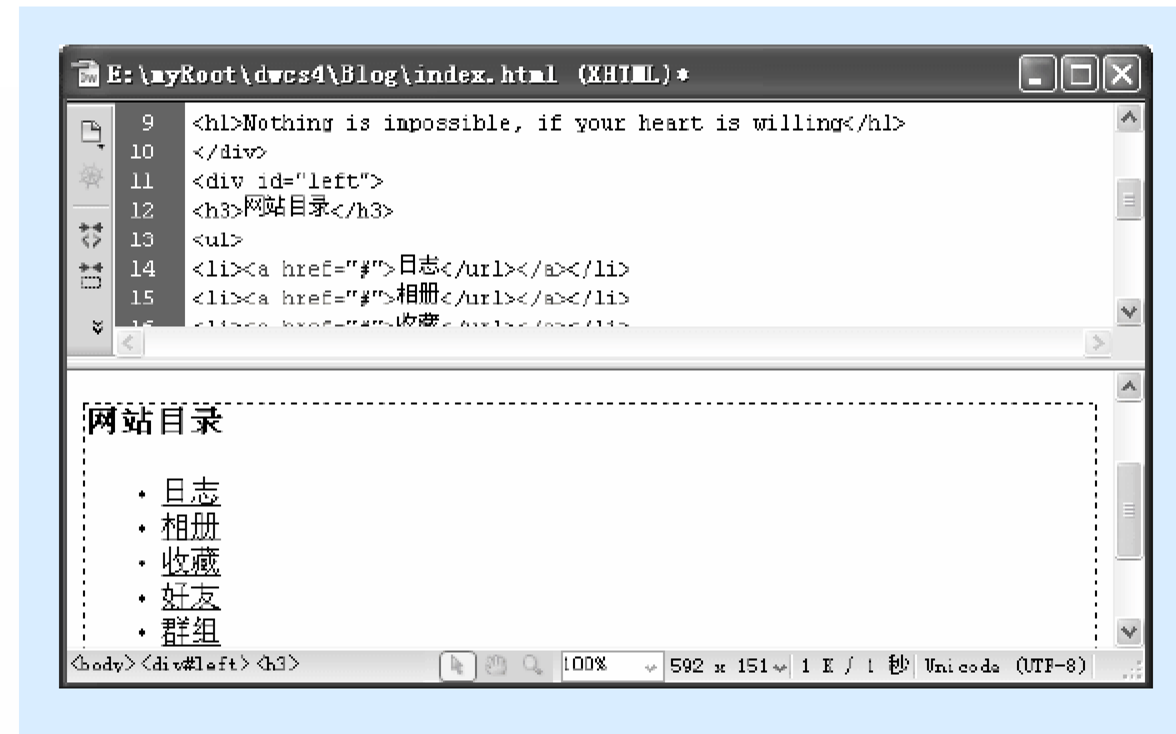


图 4-39　设置左侧栏目内容

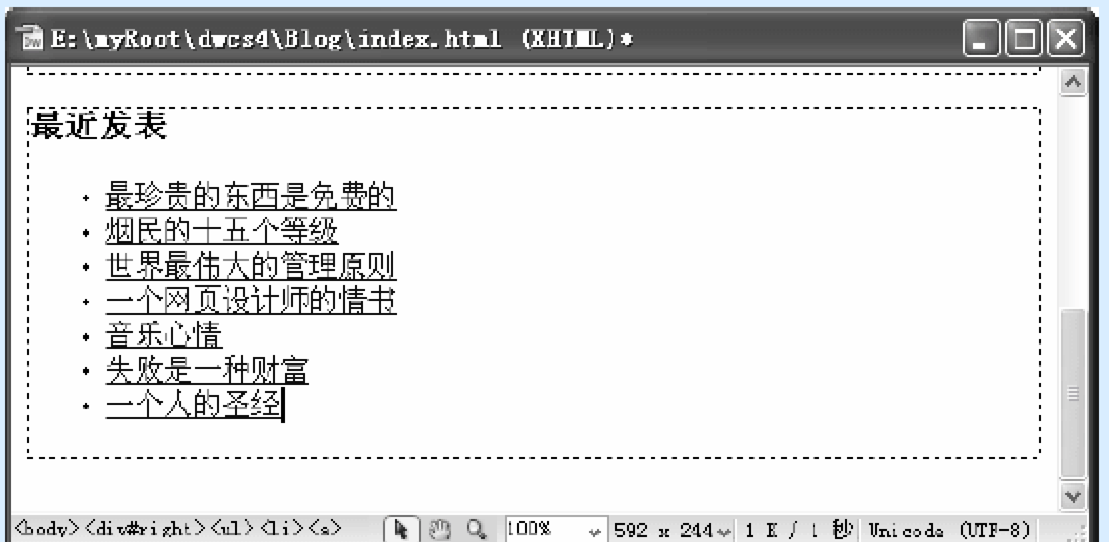


图 4-40　设置右侧栏目内容

步骤 06　在 id 为 middle 的标记中输入文章正文，标题设为 2 级，如图 4-41 所示。

步骤 07　在 id 为 footer 的标记中输入版权信息作为页脚，如图 4-42 所示。

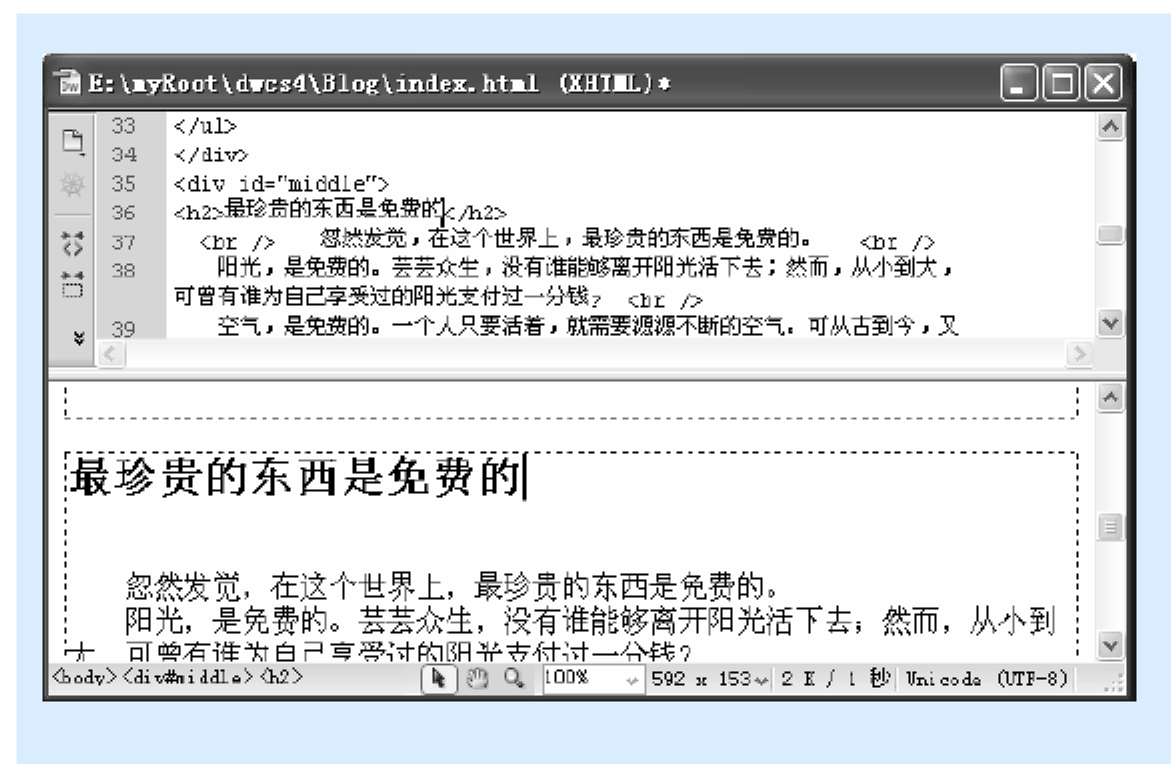


图 4-41　设置文章标题和正文内容

图 4-42　设置页脚内容

步骤 08　现在我们有了一个完整的、没有经过修饰的 XHTML 文档，下面可以运用 CSS 来控制它的布局了。首先是调整<body>，设置边距和填充都为 0，背景颜色为#BEBFC1，字体大小为 12 像素，行高为 20 像素。将文档切换到代码窗口，在<head>标记中输入如下代码(见图 4-43)：

```
<style type="text/css">
<!--
body {
    margin: 0px;
    padding: 0px;
```

```
    background-color: #BEBFC1;
    font-size: 12px;
    line-height: 20px;
}
-->
</style>
```

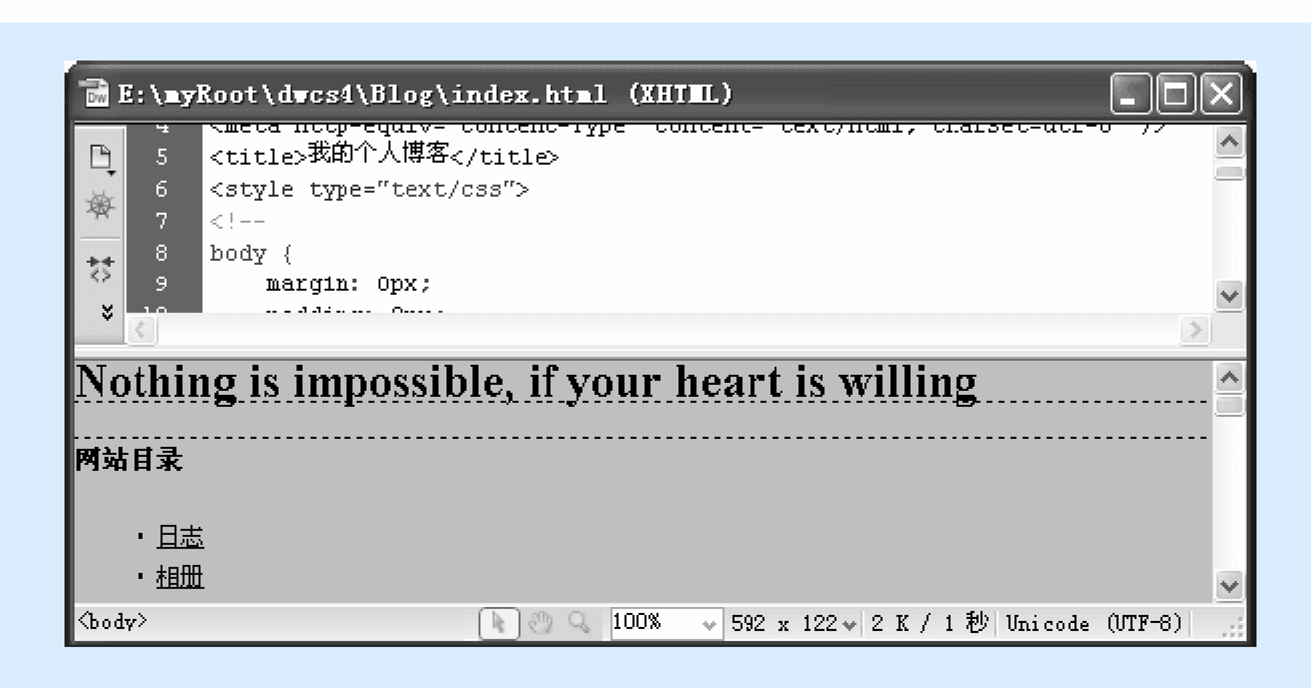


图 4-43　定义body样式

注意：有些低版本的浏览器不能识别 style 标记，这意味着低版本的浏览器会忽略 style 标记里的内容，并把 style 标记里的内容以文本方式直接显示到页面上。为了避免这样的情况发生，可以用加 HTML 注释的方式(<!--注释-->)隐藏内容而不让它显示。

步骤 09 接下来调整标题部分，用 clear 的属性确保浮动部分不会占据标题空间，标题部分的高度为 50 像素，背景颜色为#D01F3C，填充为 20 像素，并且设置字体颜色为白色和垂直居中显示。在 style 标记中加入如下代码(见图 4-44)：

```
div#header {
    clear: both;
    height: 50px;
    background-color: #D01F3C;
    padding: 20px;
    vertical-align: middle;
    color: #FFFFFF;
}
```

步骤 10 然后调整<left>使它浮动在页面左方，宽度为 150 像素，背景颜色为白色，上端距标题 20 像素，左边距 5 像素，填充 5 像素。在 style 标记中加入如下代码(见图 4-45)：

```
div#left {
    float: left;
    width: 150px;
    background-color: #FFFFFF;
    margin-top: 20px;
    padding: 5px;
    margin-left: 5px;
}
```

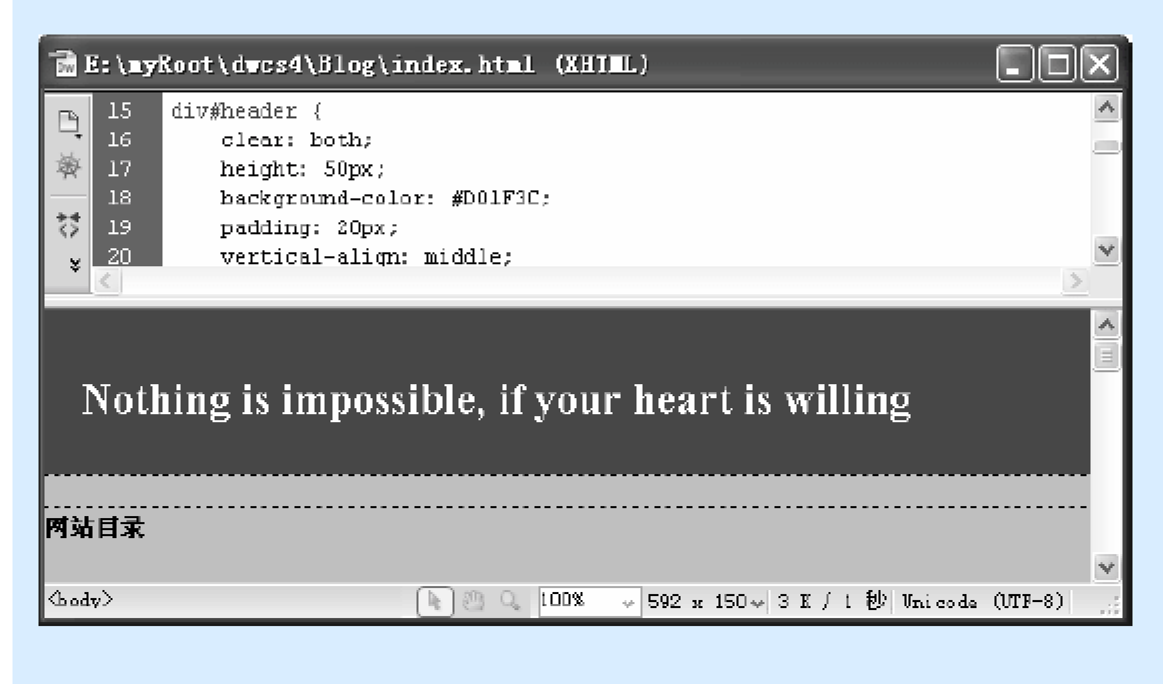


图 4-44 定义标题部分样式

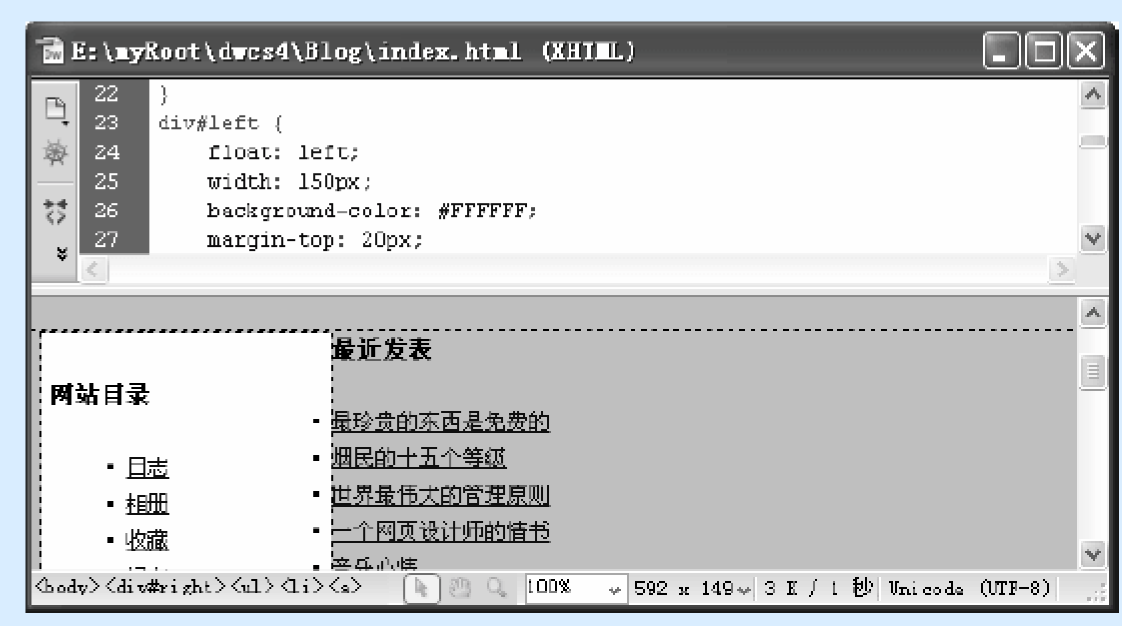


图 4-45 定义左侧栏目样式

步骤 11 调整<right>使它浮动在页面右方，宽度为 150 像素，背景颜色为#E9EBEF，上边距 20 像素，右边距 5 像素，填充 5 像素。在 style 标记中加入如下代码(见图 4-46)：

```
div#right {
    float: right;
    width: 150px;
    background-color: #E9EBEF;
    margin-top: 20px;
    margin-right: 5px;
    padding: 5px;
}
```

步骤 12 此时已经可以看初步效果了，我们再来调整 middle，使布局看起来更合理一些。设置它的边距为 0，上方填充 0 像素，下方填充 5 像素，左边和右边填充 175 像素。在 style 标记中加入如下代码(见图 4-47)：

```
div#middle {
    padding-top: 0px;
    padding-right: 175px;
    padding-bottom: 5px;
    padding-left: 175px;
    margin: 0px;
}
```



图 4-46 定义右侧栏目样式



图 4-47 定义正文样式

注意：代码中各部分出现的顺序是非常重要的。左栏和右栏 div 必须在中栏之前出现。这样才可以让这两个边栏浮动到它们的位置上(屏幕两侧)，并让中栏的内容“流”入它们之间的空间。如果浏览器在一个或者两个边栏 div 之前先发现中栏，那么中栏将占据屏幕的一侧或者两侧，这样浮动的部分就会跑到中栏的下面而不是中栏的旁边了。

步骤 13 最后调整 footer，这里 clear 声明用来确保浮动部分不会占据页脚的空间，背景颜色为#828282，填充为 10 像素，文本居中显示。在 style 标记中加入如下代码(见图 4-48)：

```
div#footer {
    clear: both;
    background-color: #828282;
    padding: 10px;
    text-align: center;
}
```

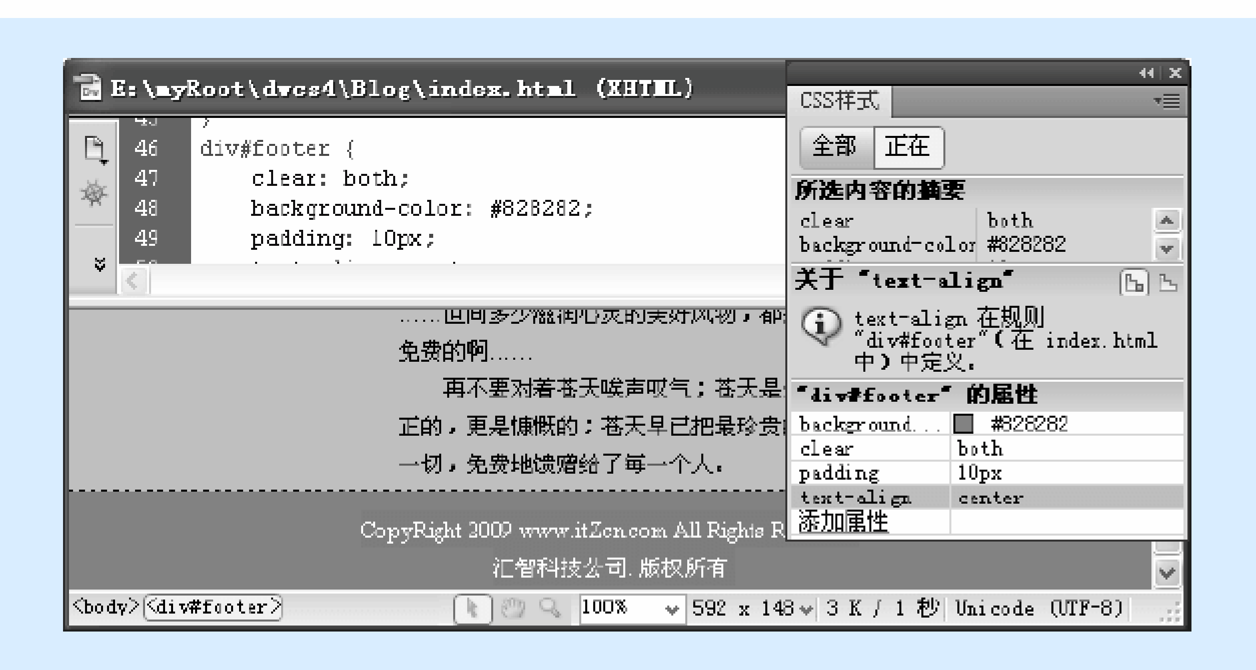


图 4-48 定义底部样式

步骤 14 一个简单使用三栏布局博客网站首页的效果制作完毕，保存文档并在浏览器中预览，最终效果如图 4-49 所示。



图 4-49 博客首页

4.8 CSS 设计规范

要想写出兼容性好，不容易出错的 CSS 样式规则，需要读者经常练习 CSS。在练习中逐渐掌握 CSS 规则对各种浏览器的兼容情况，并形成编写习惯。下面给出一些定义 CSS 规则时的建议及规范。

(1) 使用子选择器。

很多 CSS 初学者设计的样式规则多、杂、乱，而且效率非常低。其实善于使用 CSS 的子选择器可以在很大程度上提高效率，还可以减少代码量。

假设在页面中有如下 HTML 布局代码：

```
<div id=subnav>
<ul>
<li class=subnavitem> <a href=# class=subnavitem>Item 1</a></li>>
<li class=subnavitemselected> <a href=# class=subnavitemselected> Item 1
</a> </li>
<li class=subnavitem> <a href=# class=subnavitem> Item 1</a> </li>
</ul>
</div>
```

通常情况下，初学者会编写出类似如下的 CSS 样式：

```
div#subnav ul { /* Some styling */ }
div#subnav ul li.subnavitem { /* Some styling */ }
div#subnav ul li.subnavitem a.subnavitem { /* Some styling */ }
div#subnav ul li.subnavitemselected { /* Some styling */ }
div#subnav ul li.subnavitemselected a.subnavitemselected { /* Some
styling */ }
```

现在对上面的 HTML 布局进行优化，改良后的代码如下：

```
<ul id=subnav>
<li> <a href=#> Item 1</a> </li>
<li class=sel> <a href=#> Item 1</a> </li>
<li> <a href=#> Item 1</a> </li>
</ul>
```

经过改良之后，HTML 布局结构非常清晰，此时使用 CSS 的子选择器可以使代码更加简洁和容易阅读。如下所示：

```
#subnav { /* Some styling */ }
#subnav li { /* Some styling */ }
#subnav a { /* Some styling */ }
#subnav .sel { /* Some styling */ }
#subnav .sel a { /* Some styling */ }
```

(2) 省略 class 和 id 前的元素限定。

当在 HTML 页面中为元素定义一个 class 或者 id 后，如果要为该元素定义 CSS 样式，可以省略规则名称前的元素名。因为 ID 在一个页面里是唯一的，而 class 可以在页面中多次使用，所以限定某个元素是毫无意义的。例如下面的 CSS 样式：

```
div#content { /* declarations */ }
fieldset.details { /* declarations */ }
```

经过优化，省略元素名称后的 CSS 样式代码如下：

```
#content { /* declarations */ }
.details { /* declarations */ }
```

(3) 设置默认值。

通常 padding 的默认值为 0，background-color 的默认值是 transparent。但是不同的浏览器默认值可能不同。为了避免默认值不一致导致的效果错误，可以在样式表一开始就定义所有元素的 margin 和 padding 值都为 0。示例代码如下：

```
* {
margin: 0;
padding: 0;
}
```

(4) 最近优先原则。

如果在 HTML 页面针对同一个元素的 CSS 定义有很多，此时最接近元素的 CSS 定义优先级最高。假设有如下一段 HTML 代码：

```
<p class=update >您好，超级管理员</p>
```

在 CSS 文件中使用如下代码为 p 元素定义样式：

```
p {
margin: 1em 0;
font-size: 1em;
color: #333;
}
.update {
font-weight: bold;
color: #600;
}
```

上述两个规则中，.update 样式将优先被使用，因为页面中的 class 比 p 更接近内容。

(5) 区分大小写。

在 XHTML 中使用 CSS 时要注意，CSS 定义的元素名称是区分大小写的。为了避免这种错误，建议所有的定义名称都采用小写。另外，class 和 id 的值在 HTML 和 XHTML 中也是区分大小写的。所以，如果大小混合时，一定要要注意 CSS 定义名称与页面中使用的应当是一致的。

(6) 明确定义单位，除非值为 0。

只定义数值，而忽略数值的单位是很多 CSS 新手常犯的错误。在 HTML 中可以简写“width=100”，但是在 CSS 中，必须给一个准确的单位，例如“width:100px”或者“width:100em”。但是，在两种情况下可以忽略单位，即行高和 0 值。除此以外，其他值都必须紧跟单位，而且不要在数值和单位之间加空格。

(7) 不需要重复定义可继承的值。

CSS 子元素会自动继承父元素的属性值(像颜色、字体等)。因此，已经在父元素中定

义过的样式，子元素可以直接继承，不需要重复定义。但是要注意，浏览器可能用一些默认值覆盖继承的定义。

(8) 多重 class 定义。

一个元素可以同时定义多个 class。例如下面的代码定义了两个样式：

```
.one {width:200px; background:#666;}
.two {border:10px solid #F00;}
```

第一个样式设置宽度为 200 像素，背景颜色为#666；第二个样式设置 10 像素边框。

假设在 HTML 页面中使用如下代码应用这两个样式：

```
<div class="one two"></div>
```

此时的最终效果将合并两个样式的内容，即显示为宽度 200 像素、背景颜色为#666，而且边框为 10 像素。

(9) 使用 CSS 缩写。

使用缩写可以有效减少 CSS 文件的大小，更加容易阅读。

4.9 本章习题

1. 填空题

(1) Adobe 公司的________产品是功能最强大的网页设计工具。

(2) 在 IE 6 中要让所有页面元素居中，应该将 text-align 属性设置为________。

(3) 自适应布局中的________布局使用百分数来设置尺寸。

(4) 为防止弹出窗口出现水平滚动条，需将 body 元素的________属性设置为 100%。

2. 选择题

(1) 如果页面中一个 DIV 元素被设置了以下样式：

```
#div {
    width: 100px;
    height: 100px;
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
}
```

那么该元素将显示在________。

A. 页面左上角　　B. 页面右下角

C. 窗口右上角　　D. 窗口左下角

(2) 当两个垂直空白边相遇时，它们将形成一个空白边。这个空白边的高度等于两个发生叠加的空白边中的________。

A. 较小者　　B. 较大者　　C. 平均值　　D. 之和

(3) 下列不属于常见布局结构的是__________。

A. 工字布局结构　　　　B. 国字布局结构

C. 拐角型布局结构　　　D. 左右型布局结构

(4) 假设有如下的样式代码，运行后 box 的宽度是__________:

```
#box {
    width: 100px; margin: 10px; padding: 10px;
}
```

A. 140px　　B. 120px　　C. 110px　　D. 100px

3. 上机练习

根据本章学习的布局知识，在 HTML 中绘制一个如图 4-50 所示的产品介绍页面。注意页面中区块的定义，以及位置的控制。



图 4-50　示例运行的效果



第 5 章

认识HTML 5

HTML 5 从字面上很容易理解：就是 HTML 技术标准的第 5 版。大家通常所说的 HTML 5 一般是指以 HTML 5、CSS 3 和 JavaScript 为代表的最新 Web 技术(或标准)，其中 HTML 5 和 CSS 3 负责界面和内容呈现，JavaScript 则负责逻辑处理。随着 HTML 5 的不断发展，它已经成为 Web 开发的新标准，以至于浏览器对 HTML 5 的兼容性被用于区分是否能够被称为现代浏览器。作为一名 Web 开发者，必须了解 Web 开发的未来，学习使用 HTML 5 的新功能。

本章向读者介绍 HTML 5 的基础知识。通过本章的介绍，读者不仅可以熟悉 HTML 5 的发展历史和优点，还可以掌握 HTML 5 的基础语法，以及 HTML 5 中新增的与表单、标记和属性有关的内容。

本章学习目标：

- 了解 HTML 5 的发展历史
- 熟悉 HTML 5 的未来发展趋势
- 掌握 HTML 5 语法的改变
- 了解浏览器内核和常用浏览器
- 掌握浏览器的得分测试

5.1 了解 HTML 5

HTML 5 是用于取代 1999 年所制定的 HTML 4.01 和稍后 XHTML 1.0 标准的 HTML 标准版本，目前，大多数浏览器已经支持某些 HTML 5 技术。HTML 5 有两大特点：首先，强化了 Web 网页的表现性能；其次，追加了本地数据库等 Web 应用的功能。

下面来了解 HTML 5 的基本知识，包括它的发展历史、开发的三大组织、使用原因以及未来发展趋势等多个内容。

5.1.1 HTML 5 的发展历史

HTML 的历史可以追溯到很久以前，1993 年 HTML 首次以因特网的形式发布。20 世纪 90 年代的人见证了 HTML 的快速发展，从 2.0 版本到 3.2 版本和 4.0 版本，再到 1999 年的 4.01 版本，随着 HTML 的发展，W3C 掌握了对 HTML 规范的控制权。

然而，在快速发布了 HTML 的 4 个版本后，业界普遍认为 HTML 已经到了穷途末路，对 Web 标准的焦点也开始转移到了 XML 和 XHTML 上，HTML 被放在了次要位置。不过在此期间，HTML 体现了顽强的生命力，主要的网站内容还是基于 HTML 的。为了能支持新的 Web 应用，同时克服现有的缺点，HTML 迫切需要添加新功能，制定新规范。

致力于将 Web 平台提升到一个新的高度，一小组人在 2004 年成立了 WHATWG 工作组，他们创立了 HTML 5 规范，同时开始专门针对 Web 应用开发新功能。Web 2.0 这个新词正是在那个时候被发明的，Web 2.0 开创了 Web 的第二个时代，旧的静态网站逐渐让位于需要更多特性的动态网站和社交网站——这其中的新功能真的是数不胜数。

2006 年，W3C 又重新介入 HTML，并且在 2008 年 1 月 22 日发布了 HTML 5 的工作草案。2009 年，XHTML 2 工作组停止工作。又过了一年，HTML 5 已经能够解决非常实际的问题，因此，在 HTML 5 规范还未定稿的情况下，各大浏览器厂商就开始对旗下产品升级以支持 HTML 5 的新功能。这样，得益于浏览器的实验性反馈，HTML 5 规范也得到了持续的完善，它以这种方式迅速地融入到了对 Web 平台的实质性改进中。

5.1.2 开发HTML 5 的三大组织

HTML 5 草案的前身名为 Web Applications 1.0，它需要成立相应的组织，并且肯定需要有人来负责，这正是下面 3 个组织的工作。

1. WHATWG组织

WHATWG 是英文 Web Hypertext Application Technology Working Group 的缩写，中文被称为“网页超文本应用技术工作小组”，它是第一个以推动网络 HTML 5 标准为目的而成立的组织，在 2004 年，由 Apple、Mozilla、Google 和 Opera 等浏览器厂商组成。WHATWG 开发 HTML 和 Web 应用 API，同时为各浏览器厂商以及其他有意向的组织提供开放式合作。

2. W3C组织

W3C 是英文 World Wide Web Consortium 的缩写，中文被称为“万维网联盟”或者“W3C 理事会”。

W3C 组织成立于 1994 年 10 月，它是 Web 技术领域最具权威和影响力的国际中立性技术标准机构。W3C 最重要的工作是发展 Web 规范，这些规范描述了 Web 的通信协议(例如 HTML 和 XHTML)和其他的构造模块。

每一项 W3C 推荐的发展是通过由会员和受邀专家组成的工作组来完成的，在 W3C 发布某个新标准的过程中，规范是通过下面的严格程序由一个简单的理念逐步确立为推荐标准的。

(1) W3C 组织收到一份提交内容。

(2) 由 W3C 组织发布一份记录。

(3) 由 W3C 组织创建一个工作组。

(4) 由 W3C 组织发布一份工作草案。

(5) 由 W3C 组织发布一份候选的推荐。

(6) 由 W3C 组织发布一份被提议的推荐。

(7) 由 W3C 组织发布推荐。

3. IETF组织

IETF 是英文 Internet Engineering Task Force 的缩写，中文被称为“因特网工程任务组”。这个组织成立于 1985 年底，它是全球互联网最具权威的技术标准化组织，主要任务是负责互联网相关技术规范的研发和制定，当前绝大多数国际互联网技术标准出自 IETF。

5.1.3 使用HTML 5 的五大原因

HTML 5 已经被越来越多的 Web 开发者所使用，它也越来越红，那么它到底为什么这么红呢？大家可以说它天生丽质，当然它也离不开标准化组织、各大浏览器厂商及开发者的支持和追捧。下面分别从五大方面对使用 HTML 5 的原因进行分析，它们分别是技术、标准、应用、产品和媒体。

1. 技术：天生丽质

不得不说，HTML 5 天生丽质——它是跨平台的，开发简单，而且自身优势非常明显，说明如下：

- HTML 5 提高了可用性，并且改进了用户的友好体验。
- 增加了新的标记，有助于开发者定义重要的内容。
- 可以给站点带来更多的多媒体元素(例如音频和视频)。
- 可以很好地替代 Flash 和 Silverlight 技术。
- 当涉及到网站的抓取和索引的时候，对于 SEO 很友好。

- 将被大量应用于移动应用程序和游戏。
- 可移植性好。

2. 标准：天下统一

实现应用跨平台和技术和方案很多，但是只有当这个技术成为国际标准，被业界广泛接受和使用时才会显得意义非凡，影响力也会增大。W3C 领头制定 HTML 5 国际标准，原计划是 2017 年发布，现在已经提前到 2014 年，目前已经进入到最后的阶段。

3. 产品：各显神通

产品化是形成新产业的基础，令人欣喜的是多家公司能遵循同一个标准研发各种产品，推进新的产业形成。Google、苹果、Mozilla 和 Opera 等多家公司都在产品化方面做了工作。由于 Google 的大多服务都是基于 Web 的，因此，HTML 5 的成熟和广泛使用，对它绝对有好处。以 Google 为例，它的贡献包括以下几个方面：

- 在桌面操作系统已被 Windows、Mac OS 和 Linux 瓜分的情形下，Google 推出它的 Chrome 浏览器，其目标是让用户足不出 Chrome，就能完成各种需求，让 Windows 当活雷锋。
- Google 基于 Chrome 浏览器基础上，推出了 WebOS 和 Chrome OS，直接抢占 PC 桌面。
- Google 的 Android 浏览器是对 HTML 5 支持最好的手机浏览器之一。由于 HTML 5 的跨平台、跨终端的特性，相信它对 Google 在 PC 桌面方面业务部署也是重要的协同。
- Google 基于 Chrome 浏览器推出了 Web 应用商店，销售的最重要的一种应用就是 HTML 5 应用，其 Web 应用商店网址是 https://chrome.google.com/webstore。

4. 应用：众人拾柴

HTML 5 有了标准和浏览器环境后，最重要的就是应用。开发 HTML 5 应用的人越来越多，除了应用之外，HTML 5 应用 SDK 以及各种开发工具也越来越多。目前，读者所熟知的 Angrybird、植物大战僵尸、超级玛利亚，甚至 Quake 等经典游戏都是基于 HTML 5 的，而且微博和电子书报也经常会使用到 HTML 5 技术。

HTML 5 的功能非常强大，它可以和 CSS 结合，制作出漂亮的页面，也可以和 JavaScript 结合制作游戏。例如，下面列举了几个使用 HTML 5 制作的游戏应用，并且提供了这些游戏的网址。

- 植物大战僵尸：网址是 http://pvz.lonelystar.org/。
- 超级马里奥：网址是 http://www.html5china.com/html5games/mario/index.htm。
- 愤怒的小鸟：http://chrome.angrybirds.com/。
- 在线练毛笔字：http://www.theshodo.com/Write。

上面只列出了几个例子，网络上还有许多其他的例子，这里不再一一列举。如图 5-1 所示呈现了使用 HTML 5 技术实现的打方块游戏。

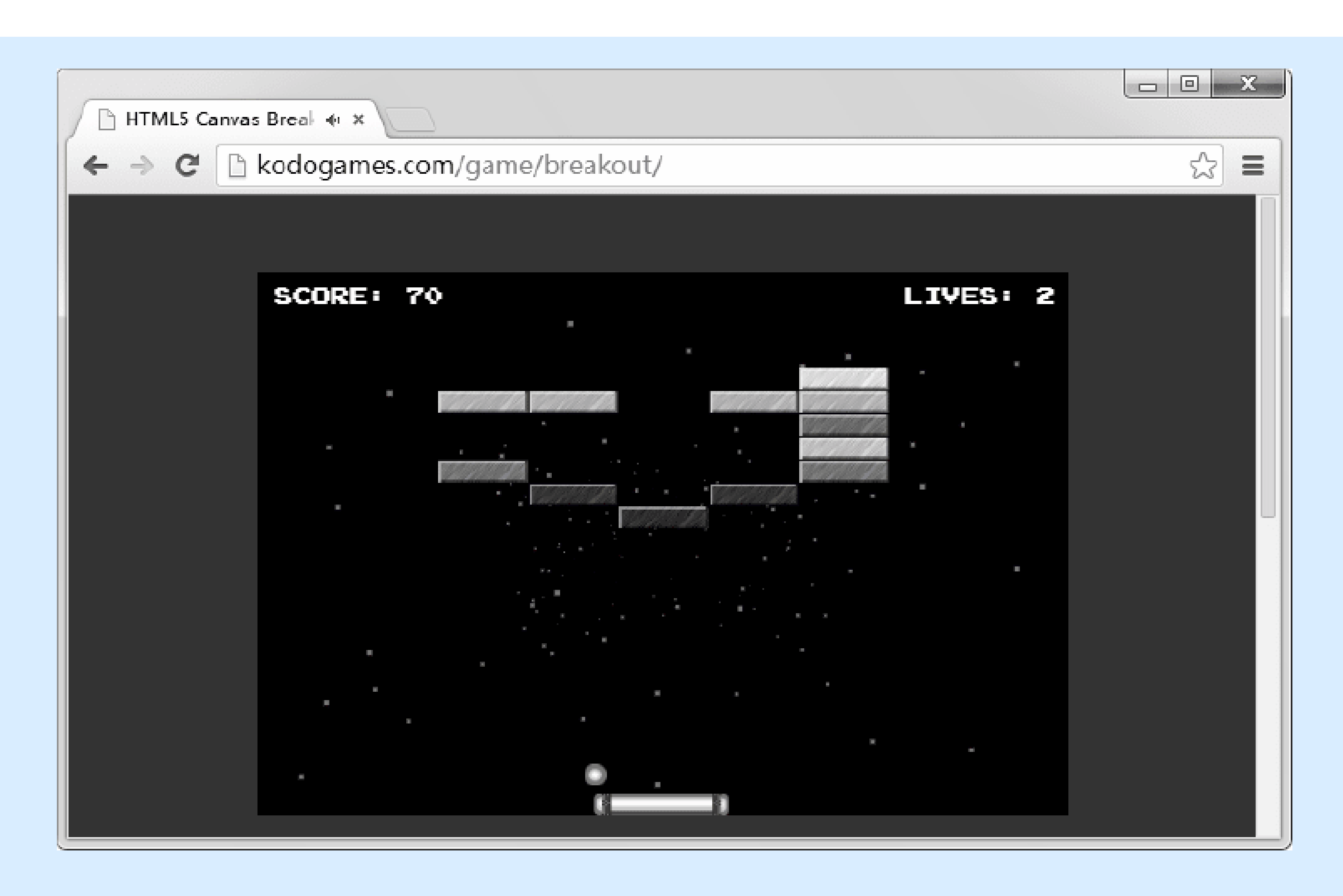


图 5-1 HTML 5 实现打方块游戏

5. 媒体：推波助澜

HTML 5 之所以这么火，除了以上几点外，当然也离不开各类媒体的推波助澜，以及各类开发者和围观者的参与。

5.1.4 HTML 5 的未来发展趋势

已经有越来越多的行业巨头在向 HTML 5 示好。除苹果、微软和黑莓之外，谷歌的 Youtube 已部分使用 HTML 5；Chrome 浏览器宣布全面支持 HTML 5；Facebook 则不遗余力地为 HTML 5 进行着“病毒式”传播。可以说，HTML5 代表了移动互联网发展的趋势，总有一天它将成为主流技术。

HTML 5 从根本上改变了开发商开发 Web 应用的方式，从桌面浏览器到移动应用，这种语言和标准都正在影响并将继续影响着各种操作平台。

1. 移动优先

在这个智能手机和平板电脑普及的时代，移动优先已经成为发展趋势，不管开发什么，都以移动为主。许多游戏开发商也将在移动 Web 应用中扮演者重要角色，移动 Web 应用优先的趋势将会持续，直到移动设备统治信息处理领域。

2. 游戏开发者领衔“主演”

移动游戏开发商是从 HTML 5 获益最多的一方，他们可利用这个平台逃脱付费游戏须向苹果支付的 30%提成。在某种程度上，游戏就是移动平台销量最好的应用，也是吸引人们购买移动设备的一个重要因素。

许多游戏开发商都被 Facebook 或者 Zynga 推动着发展，而未来的 Facebook 应用生态系统是基于 HTML 5 的，尽管在 HTML 5 平台开发出游戏非常困难，但游戏开发商却都愿意那么做。

3. 自动变化的屏幕尺寸

在 HTML 5 真正改变移动开发平台之前，必须迈出重要的一步，那就是“响应式设计”，即屏幕可以根据内容而自动调整大小。

要做好响应式设计，就必须要知道内容与屏幕之间的反馈关系。响应式最好的一个例子就是 BostonGlobe.com(观看视频)，其屏幕能够根据任何内容而调整尺寸大小。

4. 设备访问

对于许多移动开发商来说，提高设备访问能力是 HTML 5 最令人激动的革新，这意味着 Web 应用能够登录移动设备而无需做任何 PhoneGap 式打包。这就开启了另一个可能的世界，例如能与云更好地整合并提高游戏可玩性，有了 HTML 5 这个平台，开发商可以不再依赖于 Java 语言和 CSS 3 及其他程序语言。

5. 离线缓存

离线缓存是一个十分新潮的一个概念，在离线状态下，应用也能够照常动作，这是 HTML 5 充满魔力的一面。例如，亚马逊 Kindle 的云阅读器，可以通过 Firefox 6 以上版本、Chrome 11 以上版本、Safari 5 以上版本及 iOS 4 以上版本的浏览器将内容同步到所有 Kindle 系列设备，并能记忆用户在 Kindle 图书馆的一切。

6. 开发工具的成熟

在开发工具方面，除了 AppMobi 提供的工具以外，还有 Sencha 和 Appcelerator 提供的框架及 IDE 供应用开发商使用。虽然这些工具都不是特别成熟，也不如 Android 和 ISO 上的开发商框架及工作那样强大，但是它们在不断改进，将会变得越来越好用。

5.2 HTML 5 的语法

HTML 5 是 HTML 标准的下一个版本，越来越多的开发者开始使用 HTML 5 构建网站。如果同时使用 HTML 4 和 HTML 5 的话，可以发现，从头构建 HTML 5 要比 HTML 4 迁移到 HTML 5 方便得多。

下面分别从文档媒体类型、编码类型和 DOCTYPE 声明等方面介绍 HTML 5 的语法。

5.2.1 文档媒体类型

HTML 5 定义的 HTML 语法大部分都兼容 HTML 4 和 XHTML 1，但是也有一部分不兼容。大多数 HTML 文档都是保存成 text/html 媒体类型。HTML 5 为 HTML 语法定义了详细的解析规则(包括错误处理)，用户必须遵守这些规则将它保存成 text/html 媒体类型。

【例 5.1】

下面的代码是一个符合 HTML 5 语法规范的例子：

```
<!doctype html>
```

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>Example document</title>
    </head>
    <body>
        <p>Example paragraph</p>
    </body>
</html>
```

HTML 5 为 HTML 语法定义了一个 text/html 沙箱媒体类型，以便可以管理不信任的内容。其他能够用于 HTML 5 的语法是 XML，它兼容 XHTML 1，使用 XML 语法的话需要将文档保存成 XML 媒体类型，并且根据 XML 的规范需要设置命名空间，该命名空间是 http://www.w3.org/1999/xhtml。

【例 5.2】

下面的代码符合 HTML 5 中的 XML 语法规范，需要注意的是，XML 文档必须保存成 XML 媒体类型的，例如 application/xhtml+xml 或者 application/xml。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Example document</title>
    </head>
    <body>
        <p>Example paragraph</p>
    </body>
</html>
```

5.2.2 编码类型

HTML 5 的 HTML 语法中，有 3 种形式可以声明字符串的编码(encoding)类型，说明如下：

- 在传输级别上，在 HTTP 实例的 header 里设置 Content-Type 属性。
- 在文件的开头设置一个 Unicode 的 Byte Order Mark(BOM)，该字符文件的 encoding 方式提供了一个签名。
- 在文档的前 1024 个字节之前的内容里，使用带有 charset 属性的 meta 元素来声明 encoding 方式。

例如，下面的代码声明 HTML 文档是 UTF-8 格式的：

```
<meta charset="UTF-8">
```

上述代码非常简单，它替换了原来的 HTML 4 中繁琐的 meta 元素的以下声明：

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

虽然 HTML 5 兼容了 HTML 4 中的 meta 元素的语法，但是在 HTML 5 中并不推荐使用。另外，对于 HTML 5 中的 XML 语法，依然与以前的 XML 语法声明是一样的。

5.2.3 DOCTYPE声明

HTML 5 的 HTML 语法要求文档必须声明 DOCTYPE 以确保浏览器可以在标准模式下展示页面，这个 DOCTYPE 没有其他的目的，并且在 XML 中是可选项，这是因为 XML 媒体格式的文档一直就是在标准模式下处理的。

在 HTML 早期版本声明时，HTML 是建立在 SGML 基础上的，因此通过 DOCTYPE 声明时需要关联引用一个相对应的 DTD。

HTML 4 版本的 DOCTYPE 声明如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

HTML 5 与先前的版本不一样，它仅仅需要声明 DOCTYPE 就可以告诉文档启用的是 HTML 5 语法标准，浏览器会为其做剩余工作。

HTML 5 中 DOCTYPE 的声明代码如下：

```
<!DOCTYPE html>
```

5.2.4 其他内容

HTML 5 的 HTML 语法允许在文档中使用 MathML(数学标记语言)和 svg(可伸缩矢量图)元素。

例如，一个非常简单的 HTML 页面包含一个 svg 元素画出的圆。代码如下：

```
<!DOCTYPE html>
<title>SVG in text/html</title>
<p>A green circle:<svg> <circle r="50" cx="50" cy="50" fill="green"/>
</svg></p>
```

除了上述内容外，HTML 5 还支持更多复杂的组合标记，例如 svg 的 foreignObject 元素，开发者可以嵌套 MathML、HTML 或者自身嵌套。另外，HTML 5 已经原生支持 IRI (Internationalized Resource Identifiers，国际化资源标识符)了，尽管这些 IRI 只能在 UTF-8 或者 UTF-16 编码的文档中使用。

5.3 HTML 5 表单

虽然 HTML 5 还没有完全颠覆 HTML 4，它们仍然有许多相似之处，但是它们也存在着关键的不同，表单就是其中之一。本节简单了解 HTML 5 中的表单，包括输入类型、表单元素和表单属性等多个内容。

5.3.1 HTML 5 输入类型

使用 input 元素可以创建很多输入类型的控件(例如输入框、密码框、复选框、单选按

钮以及提交按钮等)，HTML 5 中新增了多个表单输入类型，这些新的输入类型提供了更好的输入控件和数据验证。

例如，表 5-1 列出 HTML 5 新增加的表单类型，且对这些表单类型进行了简单说明。

表 5-1　HTML 5 中新增的表单类型

类型名称	说　明
email	用于应该包含 E-mail 地址的输入框
url	用于应该包含 URL 地址的输入框
number	用于应该包含数值的输入框
range	用于应该包含一定范围内数值的输入框
date	选取日、月、年
month	选取月、年
week	选取周和年
time	选取时间(小时和分钟)
datetime	选取时间、日、月、年(UTC 时间)
datetime-local	选取时间、日、月、年(本地时间)
search	用于搜索域，比如站点搜索或 Google 搜索
color	用于实现一个 RGB 颜色输入
tel	从语义上实现一个电话号码的输入，常常与 pattern 属性结合使用

在 5-1 表中，date、month、week、time、datetime 和 datetime-local 都属于日期选择器，用于验证输入的日期。

表 5-1 中的输入类型的使用起来很简单，直接将<input>标记的 type 属性设置为以上类型即可。

【例 5.3】

下面的代码指定了<input>标记的不同的 type 属性值：

```
<form method="post" action="#">

    电子邮箱：<input type="email" />
    个人网页：<input type="url" />
    幸运颜色：<input type="color" /><br/><br/>
    <center><input type="submit" value="提交" /></center>

</form>
```

从上述代码中可以看出，分别向 HTML 网页中添加了 3 个<input>标记，并且将 type 属性的值指定为 email、url 和 color。

在浏览器中运行上述代码，查看效果。

如图 5-2 所示为在 Chrome 浏览器中的效果。从图中可以看出，color 类型的<input>标记向用户提供了一系列的颜色选择，用户单击颜色即可以选择。



图 5-2 HTML 5 新增的输入类型

选择用户类型完毕后，可以向电子邮箱和个人网页的输入框中输入内容，然后单击“提交”按钮，此时的效果如图 5-3 所示。



图 5-3 例 5.3 的运行效果

5.3.2 HTML 5 表单元素

在 HTML 5 中新增了一系列的元素，其中包括与表单相关的 3 个元素，它们分别是 datalist 元素、keygen 元素和 output 元素。

- datalist：该元素指定输入框的选项列表，列表通过 datalist 内的 option 元素创建。
- keygen：该元素是密钥生成器，其作用是提供一种验证用户的可靠方法。
- output：该元素用于不同类型的输出，例如计算或者脚本输出。

5.3.3 HTML 5 表单属性

在 HTML 5 中新增了一系列与表单相关的属性，下面通过两个方面简单了解一下。

1. 新的form属性

HTML 5 中新增了两个与 form 有关的属性，这两个属性分别是 autocomplete 属性和 novalidate 属性。

- autocomplete：该属性指定所有的表单控件是否应该拥有自动完成功能。如果将属性值设置为 on，表示执行自动完成功能；如果将属性值设置为 off，则表示关闭自动完成功能。
- novalidate：该属性指定在提交表单时不对表单进行有效性检查。

2. 新的input属性

除了为 form 元素增加属性外，还为 input 元素增加了一系列的属性，这些属性及其说明如表 5-2 所示。

表 5-2 HTML 5 中新增的 input 属性

属性名称	说 明
autocomplete	指定 input 输入框是否拥有自动完成功能。适用于 text、search、url、tel、email、password、datepickers、range 以及 color 类型的<input>标记
autofocus	指定页面加载后是否自动获取焦点。适用于<input>标记的所有类型
form	指定输入框所属的一个或多个表单。适用于<input>标记的所有类型
formoverrides	允许重写 form 元素的某些属性，适用于<input>标记的 submit 和 image 类型，HTML 5 表单重写属性说明如下所示。 • formaction：重写表单的 action 属性。 • formenctype：重写表单的 enctype 属性。 • formmethod：重写表单的 method 属性。 • formnovalidate：重写表单的 novalidate 属性。 • formtarget：重写表单的 target 属性
width 和 height	指定用于 image 类型的<input>标记的图像的宽度和高度
list	指定输入框的 datalist 元素的 id 值。适用于类型是 text、search、url、tel、email、datapickers、number、range 以及 color 类型的<input>标记
min、max 和 step	用于为包含数字或日期的 input 类型指定限定约束，其中 max 指定允许的最大值；min 指定允许的最小值；step 指定合法的数字间隔。适用于类型是 datapickers、number 和 range 的<input>标记
multiple	指定输入框中可以选择多个值。适用于类型是 email 和 file 的<input>标记
pattern	用于验证 input 框的模式，模式是正则表达式。适用于类型是 text、search、url、tel、email、和 password 的<input>标记
placeholder	提供一种提示，描述输入框所期待的值。适用于类型是 text、search、url、tel、email 以及 password 的<input>标记
required	指定必须在提交之前填写输入框

5.4 HTML 5 的元素

与 HTML 4 相比，HTML 5 的功能有了很大的改进，元素就是其中之一，下面分别从新增的元素、更改的元素和废除的元素三个方面进行介绍。

5.4.1 新增的元素

HTML 5 中新增了许多元素，除了 5.3.2 节介绍的新增的表单元素外，表 5-3 对其他的新增元素进行了说明。

表 5-3 HTML 5 中的新增元素

元素名称	说 明
article	定义独立于文档的内容
aside	定义页面内容之外的内容
audio	定义音频
bdi	定义文本的方向，使其脱离其周围文本的方向设置
canvas	绘制图形
command	定义命令按钮
datalist	定义下拉列表
details	定义元素的细节
embed	定义外部交互内容或插件
figure	定义媒介内容的分组，以及它们的标题
figcaption	定义 figure 元素的标题
footer	定义 section 或者 page 的页脚
header	定义 section 或者 page 的页眉
hgroup	定义有关文档中块的信息
mark	定义有记号的文本
meter	定义预定义范围内的度量
nav	定义导航链接
progress	定义任何类型的任务的进度
rp	定义如果浏览器不支持 ruby 元素显示的内容
rt	定义 ruby 注释的解释
ruby	定义 ruby 注释
section	定义文档中的节(section、区段)。例如章节、页眉、页脚或文档中的其他部分
source	定义媒介源
summary	定义 details 元素的标题

续表

元素名称	说　明
time	定义日期和时间
track	定义用在媒体播放器中的文本轨道
video	定义视频

5.4.2 更改元素

HTML 5 中对 HTML 4 已经存在的元素进行了更改，例如对某个元素添加了新的属性，或者废除了某个属性，下面介绍 HTML 5 中更改的一些常用元素。

1. html元素

html 元素是整个 HTML 文档的根元素，在 HTML 4 中的 xmlns 属性，在 XHTML 中是必需的。实际上，xmlns 属性没有任何效果，但是由于验证的原因，在把 HTML 转换为 XHTML 的过程中，是很有帮助的。在 HTML 5 中，可以不再指定 xmlns 属性，该属性是可选的。另外，HTML 5 中为 html 元素定义了一个 manifest 属性，该属性定义一个 URL，在这个 URL 中描述了文档的缓存信息。

注意：如果出于某种原因，希望定义 xmlns 属性，那么，该属性的唯一合法值是 http://www.w3.org/1999/xhtml。

2. a元素

在 HTML 4 中，a 元素既可以是超链接，也可以是锚，这取决于是否描述了 href 属性。在 HTML 5 中，a 元素是超链接，如果没有 href 属性，它仅仅是超链接的一个占位符。HTML 5 中不再支持<a>标记的 charset、coords、name、rev 和 shape 属性，新增了 type 属性和 media 属性，type 属性指定目标 URL 的 MIME 类型，仅在 href 属性存在时使用；media 属性指定目标 URL 的媒介类型，默认值是 all，仅在 href 属性存在时使用。

3. body元素

在 HTML 5 中，删除了所有 body 元素的特殊属性，这些属性是不被赞成使用的。

4. form元素

在 HTML 5 中，form 元素不再支持 HTML 4 中的 accept 属性，新增加了 autocomplete 和 novalidate 属性。

5. ol元素

对于 ol 元素，HTML 4 中不赞成使用 start 和 type 属性，但是在 HTML 5 中支持，同时新增加了 reversed 属性。

6. ul元素

在 HTML 4 中，ul 元素的 compact 和 type 属性不被赞成使用，在 HTML 5 中，不再支持这两个属性。

7. table元素

HTML 5 中只支持<table>标记的 border 属性，并且只允许使用值为空或者 1，不再支持其他的属性。

8. style元素

scoped 属性是 HTML 5 中的新属性，它允许开发者为文档的指定部分定义样式，而不是整个文档。如果使用 scoped 属性，那么所指定的样式只能应用到 style 元素的父元素及其子元素中。

9. script元素

在 HTML 5 中，<script>标记的 type 属性是必需的，但是该属性在 HTML 5 中是可选的。同时，HTML 5 中新增加了一个 async 属性，该属性指定异步执行脚本(仅适用于外部脚本)。

10. input元素

在 HTML 4 中，align 属性已经被废弃，HTML 5 不再支持该属性，如果要设置对齐样式，那么需要在 CSS 中进行设置。HTML 5 对<input>标记的 type 属性添加了多个属性值，这些属性值及其说明可以参考表 5-2。

注意：除了上面介绍的 10 个元素外，HTML 5 中还对其他的一些元素进行了或多或少的改进，但是其他的元素并不经常被用到，因此这里不再详细进行解释和说明。

5.4.3 废除的元素

HTML 5 在增加或者修改元素的同时，也对 HTML 4 中的一些元素进行了删除。例如，表 5-4 中列出了 HTML 5 中废除的元素，这些元素在 HTML 5 中不再提供支持。

表 5-4 HTML 5 中的废除元素

元素名称	说 明
acronym	定义首字母缩写
applet	定义嵌入的 applet。可以使用 object 来代替
basefont	定义文档中所有文本的默认颜色、大小和字体。可以使用 CSS 样式代替
big	定义大号文本
center	定义居中的文本

续表

元素名称	说　明
dir	定义目录列表
font	定义文本的字段、颜色和大小。可以使用 CSS 样式代替
frame	定义子窗口(框架)
frameset	定义框架集
isindex	定义单行的输入框
noframes	向浏览器显示无法处理框架的提示文本
s 和 strike	定义加删除线的文本
tt	定义打字机文本
u	定义下划线文本
xmp	定义预格式文本

5.5　HTML 5 的属性

HTML 5 在对元素增加、废除和改进的同时，也对属性进行了操作。在 5.3.3 节已经介绍了 HTML 5 中新增的表单属性，除了这些属性外，HTML 5 中还增加了其他的属性，下面从标准属性和事件属性两方面进行说明。

5.5.1　标准属性

标准属性是指 HTML 中的元素都支持的属性，只有极少数的元素是个例外。例如，在表 5-5 中列出了 HTML 5 中新增的元素的标准属性，并且对这些属性进行了简单说明。

表 5-5　HTML 5 中新增的标准属性

属性名称	说　明
contenteditable	指定是否允许用户编辑内容
contextmenu	指定元素的上下文菜单
data-yourvalue	创建者定义的属性，HTML 文档的创作者可以定义他们自己的属性。定义属性时必须以 data-开头
draggable	指定是否允许用户拖动元素
hidden	指定元素是无关的，被隐藏的元素不会显示
item	用于组合元素
spellcheck	指定是否必须对元素进行拼写或者语法检查
subject	指定元素对应的项目

上面表 5-5 列举的属性中，并不是每一个属性都会被使用到，有些属性并不常用。下面通过一个示例演示 contenteditable 属性的使用。

【例 5.4】

contenteditable 属性指定是否允许用户编辑内容，它几乎适用于所有的元素。下面通过<pre>标记显示一些个人资料，并且为该标记指定 contenteditable 属性。HTML 页面的相关代码如下：

```
个人资料信息：
<pre id="personinfo" contenteditable>
我叫李想容，今年 24，幸运数字是 8。
          职业：淘宝客服
          爱好：购物、唱歌、跳舞和画画
  最喜欢的歌曲：抒情、充满正能量的歌曲
最喜欢的电视剧：喜欢古装剧
别人对我的评价：
</pre>
```

在浏览器中运行上述代码查看效果，初始效果如图 5-4 所示。



图 5-4　例 5.4 的初始效果

从例 5.4 的代码中可以看出，为<pre>标记添加了 contenteditable 属性，它表示 pre 元素中的内容可以编辑。将鼠标移动到 pre 元素所在处并单击，此时的区域是可编辑的，可以向编辑区域添加或修改内容，如图 5-5 所示。



图 5-5　编辑区域内容

5.5.2 事件属性

HTML 4 增加了通过事件触发浏览器中行为的能力，例如，当用户点击某个元素时，启动一段 JavaScript 脚本。HTML 5 中增加了新的事件属性，下面对 Window 事件属性、表单事件属性、键盘事件属性、鼠标事件属性和媒介事件属性分别进行说明。

1. Window事件属性

Window 事件是指 Window 对象触发的事件。Window 事件属性适用于 body 元素，例

如 onload。HTML 5 中为 body 元素增加了多个 Window 事件属性，这些事件属性及其说明如表 5-6 所示。

表 5-6　HTML 5 中新增的 Window 事件属性

Window 事件属性	说　明
onafterprint	在打印文档之后运行脚本
onbeforeprint	在文档打印之前运行脚本
onbeforeonload	在文档加载之前运行脚本
onerror	当错误发生时运行脚本
onhaschange	当文档改变时运行脚本
onmessage	当触发消息时运行脚本
onoffline	当文档离线时运行脚本
ononline	当文档上线时运行脚本
onpagehide	当窗口隐藏时运行脚本
onpageshow	当窗口可见时运行脚本
onpopstate	当窗口历史记录改变时运行脚本
onredo	当文档执行再执行操作(redo)时运行脚本
onresize	当调整窗口大小时运行脚本
onstorage	当文档加载时运行脚本
onundo	当 Web Storage 区域更新时(存储空间中的数据发生变化时)运行脚本
onunload	当用户离开文档时运行脚本

2. 表单事件属性

表单事件是由 HTML 表单内部的动作触发的事件，例如 onblur、onchange、onfocus 和 onsubmit 等都属于表单事件属性，这些表单事件属性适用于所有的 HTML 5 元素，但是这些事件属性最常用于表单元素中。HTML 5 中不再支持 onreset 事件属性。表 5-7 列出了 HTML 5 中新增的表单事件属性。

表 5-7　HTML 5 中新增的表单事件属性

表单事件属性	说　明
oncontextmenu	当触发上下文菜单时运行脚本
onformchange	当表单改变时运行脚本
onforminput	当表单获得用户输入时运行脚本
oninput	当元素获得用户输入时运行脚本
oninvalid	当元素无效时运行脚本

3. 键盘事件属性

键盘事件是由键盘触发的事件，onkeydown、onkeypress 和 onkeyup 都是键盘事件属

性，适用于所有的 HTML 5 元素。在 HTML 5 中，并没有增加新的键盘事件属性，因此这里不再详细说明。

4. 鼠标事件属性

鼠标事件是指由鼠标或相似的用户动作触发的事件，例如 onclick、ondbclick 和 onmousedown 等都属于鼠标事件属性，这些属性适用于所有的 HTML 5 元素。在 HTML 5 中，新增加了 9 个鼠标事件属性，这些属性及其说明如表 5-8 所示。

表 5-8　HTML 5 中新增的鼠标事件属性

鼠标事件属性	说　明
ondrag	当拖动元素时运行脚本
ondragend	当拖动操作结束时运行脚本
ondragenter	当元素被拖动至有效的拖放目标时运行脚本
ondragleave	当元素离开有效拖放目标时运行脚本
ondragover	当元素被拖动至有效拖放目标上方时运行脚本
ondragstart	当拖动操作开始时运行脚本
ondrop	当被拖动元素正在被拖放时运行脚本
onmousewheel	当转动鼠标滚轮时运行脚本
onscroll	当滚动滚动元素的滚动条时运行脚本

5. 媒介事件属性

媒介事件是由视频、图像以及音频等媒介元素触发的事件。媒介事件属性适用于所有 HTML 5 元素，但是在媒介元素(例如 audio 元素和 video 元素)中最为常用。HTML 5 中增加了多个媒介事件属性，如表 5-9 所示。

表 5-9　HTML 5 中新增的媒介事件属性

媒介事件属性	说　明
oncanplay	当媒介能够开始播放但可能因缓冲而需要停止时运行脚本
oncanplaythrough	当媒介能够无需因缓冲而停止即可播放至结尾时运行脚本
ondurationchange	当媒介长度改变时运行脚本
onemptied	当媒介资源元素突然为空时(网络错误、加载错误等)运行脚本
onended	当媒介已抵达结尾时运行脚本
onerror	当在元素加载期间发生错误时运行脚本
onloadeddata	当加载媒介数据时运行脚本
onloadedmetadata	当媒介元素的持续时间以及其他媒介数据已加载时运行脚本
onloadstart	当浏览器开始加载媒介数据时运行脚本
onpause	当媒介数据暂停时运行脚本
onplay	当媒介数据将要开始播放时运行脚本

续表

媒介事件属性	说　明
onplaying	当媒介数据已开始播放时运行脚本
onprogress	当浏览器正在取媒介数据时运行脚本
onratechange	当媒介数据的播放速率改变时运行脚本
onreadystatechange	当就绪状态(ready-state)改变时运行脚本
onseeked	当媒介元素的定位属性不为真且定位已结束时运行脚本
onseeking	当媒介元素的定位属性为真且定位已开始时运行脚本
onstalled	当取回媒介数据过程中(延迟)存在错误时运行脚本
onsuspend	当浏览器已在取媒介数据但在取回整个媒介文件之前停止时运行脚本
ontimeupdate	当媒介改变其播放位置时运行脚本
onvolumechange	当媒介改变音量或当音量被设置为静音时运行脚本
onwaiting	当媒介已停止播放但打算继续播放时运行脚本

5.6　支持 HTML 5 的浏览器

只有技术标准是远远不够的，更重要的是浏览器产品遵循和支持这个标准，这是因为 HTML 5 应用需要类似浏览器这样的特定应用执行环境。下面从浏览器内核开始介绍，然后再介绍支持 HTML 5 的一些常用的浏览器。

5.6.1　浏览器内核

浏览器最重要或者核心的部分是 Rendering Engine，可以翻译为“解释引擎”，不过通常会将其称为“浏览器内核”，负责对网页语法的解释(例如 HTML 和 JavaScript)并渲染显示网页。因此，通常所说的“浏览器内核”就是浏览器所采用的渲染引擎，它决定了浏览器如何显示网页的内容以及页面的格式信息。

不同的浏览器内核对网页编写语法的解释也有不同，因此同一网页在不同内核的浏览器中的渲染显示效果可能不同，因此，网页编写者需要在不同内核的浏览器中测试网页显示效果。下面介绍几种常用的浏览器内核，它们分别是 Trident 内核、Gecko 内核、Presto 内核、WebKit 内核以及 Blink 内核。

1. Trident内核(IE内核)

Trident 内核程序在 1997 年的 IE 4 中首次被采用，是微软在 Mosaic 代码的基础上修改而来的，并沿用到 IE 11。Trident 是一款开放的内核，其接口内核设计得相当成熟，因此才有许多采用 IE 内核而非 IE 的浏览器涌现。另外，为了方便，许多人将 Trident 内核称为 IE 内核。

Trident 内核的常用浏览器包括 IE6、IE7、IE8(Trident 4.0)、IE9(Trident 5.0)、IE10

(Trident 6.0)；360 安全浏览器、傲游浏览器(傲游 1.x、2.x 为 IE 内核，3.x 为 IE 与 WebKit 双核)、百度浏览器(早期版本)、世界之窗浏览器(最初为 IE 内核，2013 年采用 Chrome+IE 内核)以及瑞星安全浏览器等。

其中部分浏览器的新版本是“双核”甚至是“多核”，其中一个内核是 Trident，然后再增加一个其他内核。国内的厂商一般把其他内核叫作“高速浏览模式”，而 Trident 则是“兼容浏览模式”，用户可以来回切换。

2. Gecko内核(Firefox内核)

它是 Netscape 6 开始采用的内核，后来的 Mozilla Firefox(火狐浏览器)也采用了该内核。Gecko 的特点是代码完全公开，可开发程度很高，全世界的程序员都可以为其编写代码增加功能。由于 Gecko 是一个开源内核，因此受到许多人的青睐，使用 Gecko 内核的浏览器也有很多，这也是 Gecko 内核虽然年轻但市场占有率能够迅速提高的重要原因。

Gecko 内核常见的浏览器包括 Mozilla Firefox、Mozilla SeaMonkey、Epiphany(早期版本)、Flock(早期版本)和 K-Meleon 等。

3. Presto内核(Opera前内核)

Presto 内核在 2003 年的 Opera 7 中首次被使用，其特点就是渲染速度的优化达到了极致，然而代价是牺牲网页的兼容性。Presto 实际上是一个动态内核，与前几个内核的最大区别在于脚本处理。使用 Presto 的除了 Opera 浏览器外，还有 NDSBrowser、Wii Internet Channle 和 Nokia 770 网络浏览器等，这在很大的程度上限制了 Presto 的发展。

4. WebKit内核

Presto 是 Opera 浏览器的前内核，Opera 12.6 及更早版本曾经采用的内核，现在已经停止并废弃。Opera 现已改用 Google Chrome 的 WebKit 内核，它是一个开源的浏览器内核，其优势在于高效稳定、兼容性好、且源码结构清晰、易于维护。WebKit 引擎包含 WebCore 排版引擎和 JavaScriptCore 解析引擎，这两个引擎都是从 KDE 的 KHTML 及 KJS 引擎衍生而来的。

Chrome 浏览器、Safari 浏览器、Android 默认浏览器、360 极速浏览器以及搜狗浏览器高速模式都使用 WebKit 作为内核(在脚本理解方面，Chrome 使用自己研发的 V8 引擎)。WebKit 内核在手机的应用也十分广泛，例如 Google 的手机 Gphone、Apple 的 iPhone、Nokia 的 Series 60 Browser 等所使用的内核引擎，都是基于 WebKit 的。

5. Blink内核(Google的未来内核)

2013 年 4 月 3 日，谷歌在 Chromium Blog 上发表博客，称将与苹果的开源浏览器核心 Webkit 分道扬镳，在 Chromium 项目中自主研发 Blink 渲染引擎(即浏览器核心)，内置于 Chrome 浏览器中。

5.6.2 常用的浏览器

目前，许多主流的浏览器都提供了对 HTML 5 技术的支持。前面介绍了常用的浏览器

内核，这里将介绍支持 HTML 5 的常用的 7 款浏览器，其中 Chrome 浏览器和 Firefox 浏览器最常用。

1. Chrome浏览器

Google Chrome 又称为 Google 浏览器，是由 Google(谷歌)公司开发的网页浏览器。该浏览器基于其他开源软件撰写，包括 WebKit，其目标是提升稳定性、速度和安全性，并创造出简单且有效率的使用者界面。

Chrome 浏览器支持多标签浏览，每一个选项卡面都在独立的“沙箱”内运行，在提高安全性的同时，一个选项卡面的崩溃不会导致其他标签被关闭。另外，Chrome 浏览器基于更强硬的 JavaScript V8 引擎，这是当前其他 Web 浏览器所无法实现的。

2. Firefox浏览器

Mozilla Firefox，中文名通常称为“火狐”或“火狐浏览器”，它是一个自由的、开放源码的网页浏览器，使用 Gecko 引擎(非 IE 内核)，支持多种操作系统(例如 Windows、Mac 和 Linux)。

Firefox 浏览器除了体积小、速度快外，还有其他一些高级特征。例如安全性高、稳定性好；运行速度快，占用资源少；个性化十足；功能完善；方便开发和调试等。

3. Opera浏览器

Opera 浏览器即欧朋浏览器，它是一个来自挪威的一个极为出色的浏览器，具有速度快、节省系统资源、订制能力强、安全性高以及体积小等特点，目前已经是最受欢迎的浏览器之一。

Opera 浏览器创始于 1995 年 4 月，它支持多种操作系统，例如 Windows、Linux、Mac、FreeBSD、Solaris 和 OS/2 等。另外，Opera 还有手机用的版本，例如在 Android 手机上安装的 Opera Mobile。

4. IE浏览器

Internet Explorer 原称 Microsoft Internet Explorer 和 Windows Internet Explorer，简称 IE，这是微软公司推出的一款网页浏览器。目前，IE 的最新版本是 11，它的内核是 Trident 7.0。

5. Safari浏览器

苹果公司推出一款兼容微软 Windows 操作系统的 Safari 网络浏览器，试图与它强大的对手分庭抗礼。Safari 浏览器已经支持 Windows 平台，但是与运行在 Mac OSX 上的 Safari 相比，有些功能出现丢失。该浏览器也是 iPhone 手机、iPodTouch 和 iPad 平板电脑中指定的默认浏览器。

6. 傲游浏览器

傲游浏览器(傲游 1.x、2.x 为 IE 内核，3.x 为 IE 与 WebKit 双核)是一款多功能、个性化多标签浏览器。目前，傲游浏览器的最新版本为 4，与前几个版本相比，它的功能更加

强大，增加了多标签浏览、智能加速、智能填表、在线收藏服务、Feed 订阅、屏幕截图、傲游下载，以及过滤包和多语言支持等多个内容。

7. 搜狗浏览器

搜狗浏览器被更名为“搜狗高速浏览器”，这是目前互联网上最快速、最流畅的新型浏览器，与拼音输入法、五笔输入法等产品一同成为用户高速上网的必备工具。搜狗高速浏览器拥有国内首款“真双核”(WebKit 和 IE)引擎，采用多级加速机制，能大幅度提高用户的上网速度。

5.7 实战——Chrome 浏览器的安装和测试

从前面的介绍中可以看到，目前，已经有许多浏览器支持 HTML 5 技术。由于 HTML 5 标准尚未定稿，导致各种浏览器产品对 HTML 5 的支持程度差异较大。因此，本节以 Chrome 浏览器为例进行安装，安装完毕后在测试网站上对该浏览器进行测试。

Chrome 浏览器的安装和测试步骤如下。

步骤 01 下载最新版本的 Chrome 浏览器，这里下载的版本是 v33.0.1750.149。

步骤 02 找到下载的 Chrome 浏览器的安装文件并双击，这时弹出如图 5-6 所示的安装对话框。



图 5-6 安装Chrome浏览器

步骤 03 Chrome 浏览器安装完毕后的界面如图 5-7 所示。

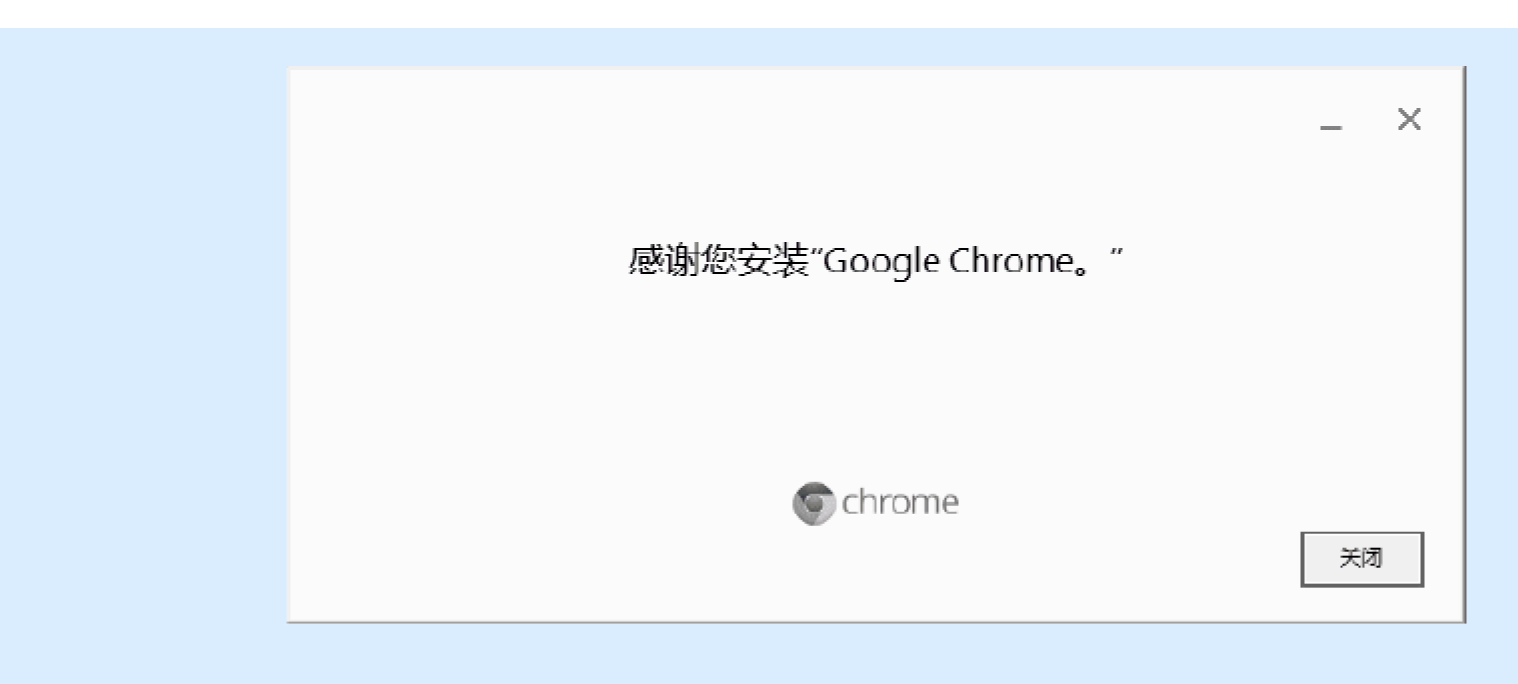


图 5-7 Chrome浏览器安装完成

提示：由于各种浏览器对 HTML 5 的支持程度差异较大，因此本书在介绍 HTML 5 时会使用到多个浏览器进行比较。本节只介绍 Chrome 浏览器的安装和测试，读者可以安装其他的浏览器(例如 Firefox 浏览器和 Opera 浏览器)，安装成功后再进行测试。

步骤 04　浏览器安装完毕后，需要对其进行测试，测试某一浏览器对 HTML 5 的支持程度很简单，直接在地址栏中输入 http://www.html5test.com 网址进行访问，如图 5-8 所示。

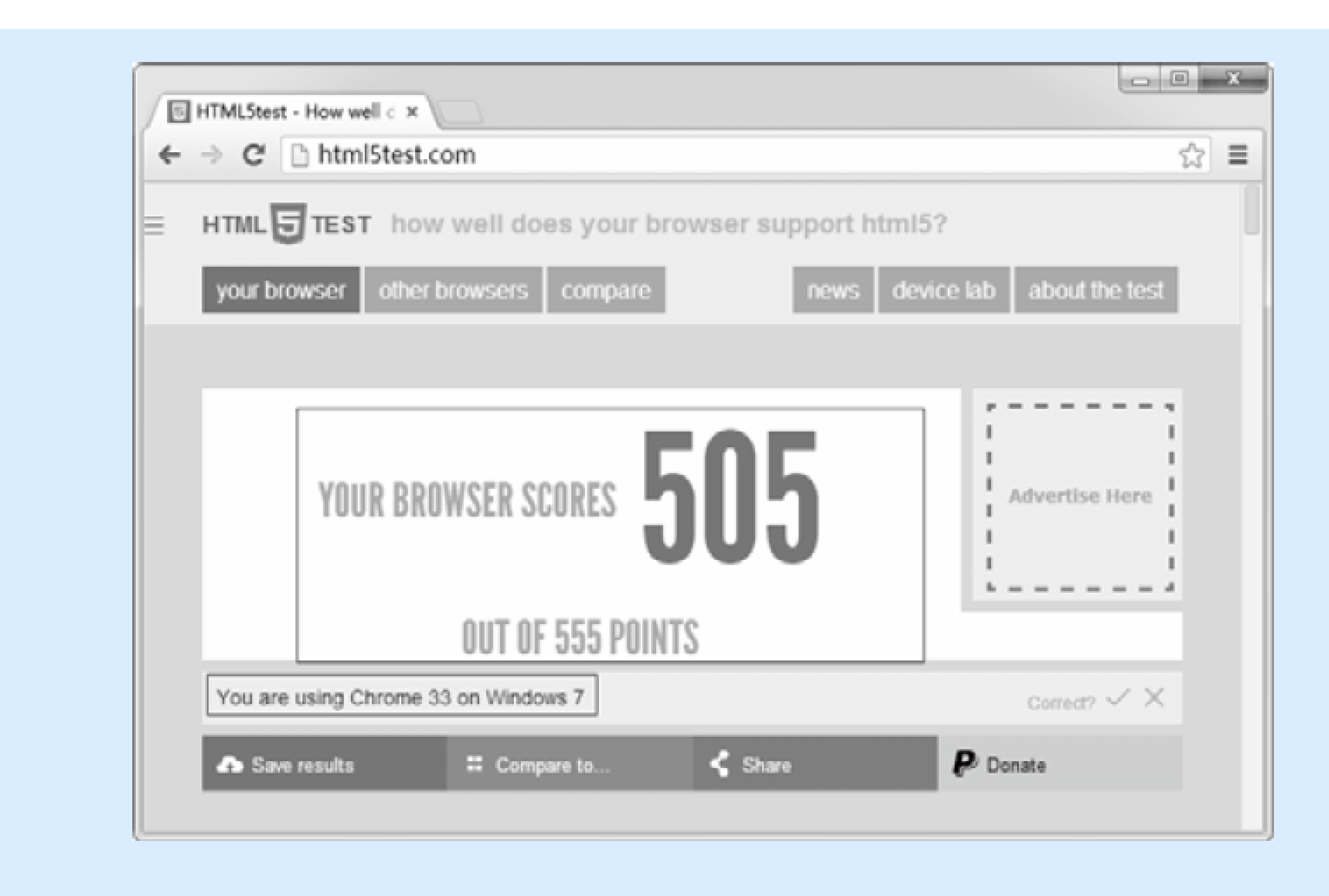


图 5-8　Chrome浏览器的得分

从图 5-8 中可以看出，当前使用的 Chrome 浏览器的得分是 505 分(总分 555 分)，而且当前使用的是 Windows 7 操作系统，当前浏览器版本是 33。

步骤 05　向下拖动图 5-8 中的滚动条，查看浏览器对 HTML 5 的具体支持情况，例如图 5-9 显示了 Chrome 浏览器对 Elements(元素)和 Audio 元素的支持情况。

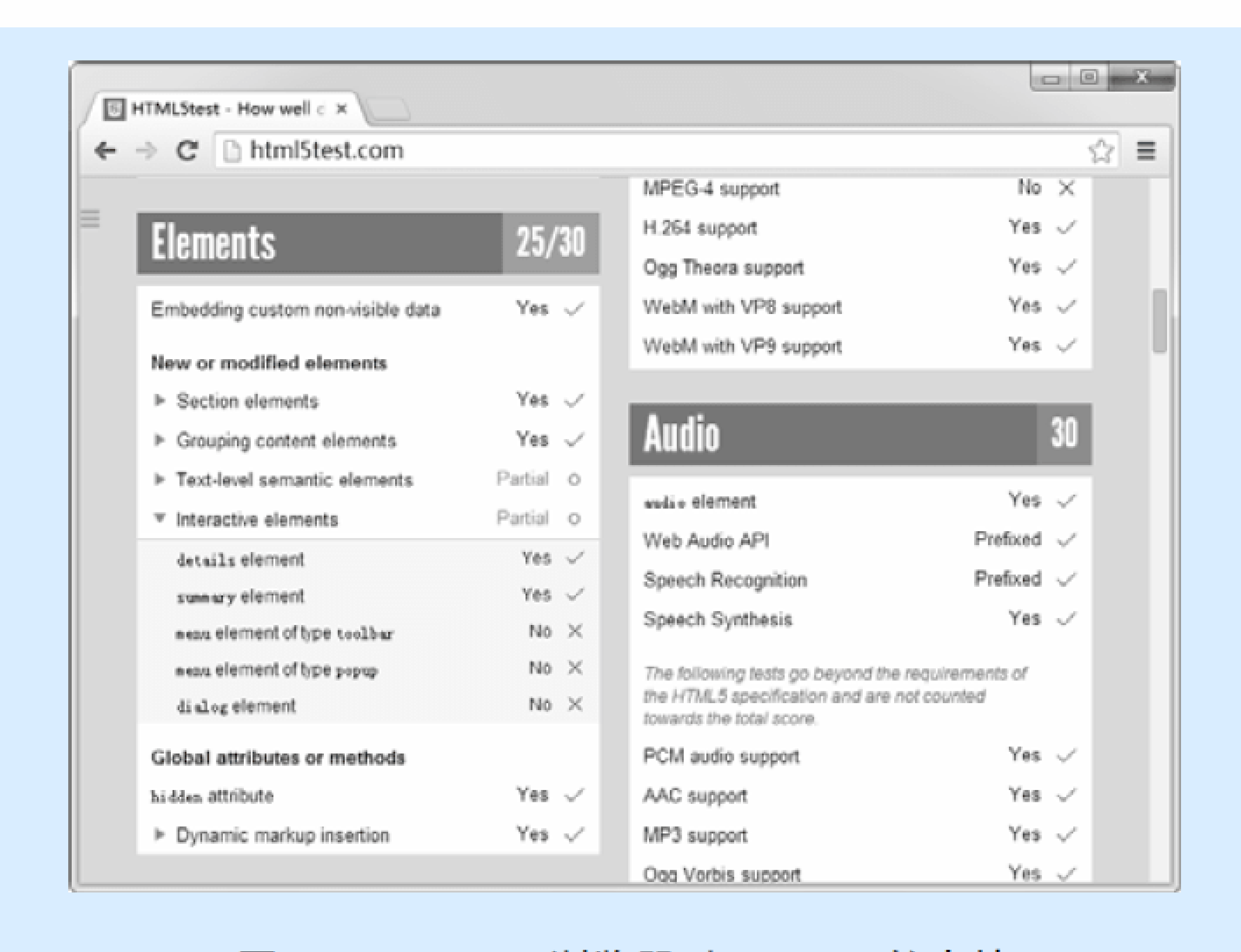


图 5-9　Chrome浏览器对HTML 5 的支持

从图 5-9 中可以看出，对于 Elements 部分的内容来说，得分是 25 分(共 30 分)，单击打开某些菜单项，发现该浏览器不支持部分元素，例如 dialog 元素和 menu 元素等。对于 Audio 元素来说，得分 30 分，这表示当前浏览器支持 audio 的所有功能，包括 AAC、MP3 和 Ogg 格式文件的支持。

提示：图 5-9 只是显示了 Chrome 浏览器对 HTML 5 的部分截图，在该测试网站还可以查看对 video 元素、输入内容、离线应用、文件以及 3D 和 2D 图形的支持情况，这里不再显示具体的效果，读者可以登录该网站进行测试和查看。

步骤 06 单击图 5-8 中的 other browsers 菜单项，可以查看其他浏览器的得分以及先前旧版本的得分，如图 5-10 所示。

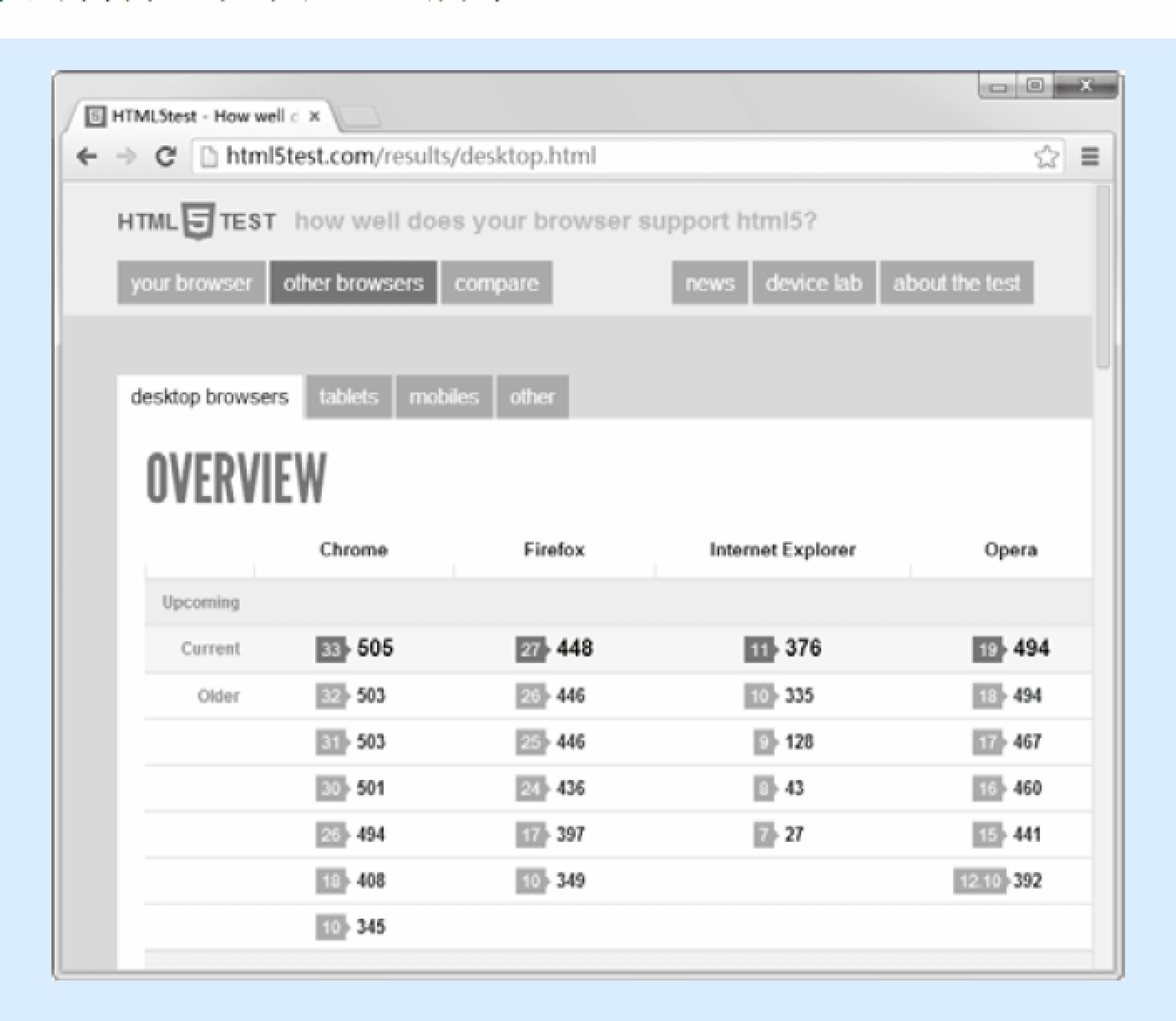


图 5-10　其他浏览器的得分

图 5-10 对 Chrome 浏览器、Firefox 浏览器、IE 浏览器以及 Opera 浏览器进行比较，Current 表示当前最新版本的得分，Older 表示旧版本的得分。

步骤 07 拖动图 5-10 的工具条查看其他信息，或者单击该测试网站的其他选项查看，这里不再显示具体的操作效果。

5.8 本 章 习 题

1. 填空题

(1) 开发 HTML 5 的三大组织是 WHATWG、__________和 IETF。

(2) HTML 5 中使用 DOCTYPE 的声明语法是__________。

(3) 在 HTML 5 新增的输入类型中，__________类型用于应该包含数值的输入框。

(4) __________和 novalidate 是 HTML 5 新增的与表单有关的两个属性。

(5) 在 HTML 5 新增的标准属性中，__________属性指定元素的上下文菜单。

2. 选择题

(1) HTML 5 中新增的表单元素不包括__________。

A. output B. header C. datalist D. keygen

(2) 在 HTML 5 新增的表单属性中，__________属性用于验证输入框的模式。

A. multiple B. autofocus C. pattern D. form

(3) 下面的选项中，__________不是 HTML 5 新增加的元素。

A. command B. meter C. time D. input

(4) 下面的选项中，__________是 HTML 5 中废除的元素。

A. input B. frameset C. meta D. progress

(5) 下面的选项中，__________不是 HTML 5 新增的表单事件属性。

A. onerror B. onforminput C. oncontextmenu D. oninput

(6) IE 浏览器使用的内核是__________。

A. Gecko B. Presto C. WebKit D. Trident

3. 上机练习

(1) 安装 Firefox 浏览器并测试得分

读者可以从网站下载最新版本的 Firefox 浏览器并进行安装，安装完毕后测试该浏览器的得分。

(2) 安装 Opera 浏览器并测试得分

读者可以从网站下载最新版本的 Opera 浏览器并进行安装，安装完毕后测试该浏览器的得分。



第 6 章

HTML 5 快速入门

HTML 5 的新增元素和属性是它的一大亮点，这些新增元素使文档结构更加清晰明确，容易阅读，属性则有助于读者实现更强大的功能。根据 HTML 5 新增元素的使用情况和语义，可以将它们进行不同的分类。有些元素的定义很模糊，以 header 元素来说，它既可以是结构元素，也可以作为语义元素，可以将该元素放到任意一种类型中，这也是为什么在不同的参考资料中，同一个元素所属分类不同的原因。

本章将 HTML 5 中的新增元素分为结构元素、分组元素、语义元素、交互元素以及音频和视频元素，除了介绍这些元素外，还会介绍 HTML 5 中常用的几种标准属性。通过本章的学习，读者可以熟练地使用这些属性来构建网页。

本章学习目标：

- 掌握 HTML 5 中的结构、分组、文本语义元素
- 掌握 meter 和 progress 元素
- 熟悉 details 和 summary 元素
- 掌握 video、audio 如何显示视频/音频
- 掌握 hidden 属性的使用
- 熟悉 contenteditable、spellcheck 属性的使用

6.1 结 构 元 素

在 HTML 5 中，为了使文档的结构更加清晰明确，逻辑思路更加清晰，追加了一些与页眉、页脚、内容区块等文档结构关联的结构元素。

6.1.1 header元素

HTML 5 新增的 header 元素是一种具有引导和导航作用的结构元素，用于定义文档的页眉(介绍信息)。header 元素通常用来放置整个页面或页面内的一个内容区块的标题，也可以包含网站 Logo 图片、数据表格和搜索表单等内容。

整个页面的标题应该放在页面的开头，它的使用与其他元素一样。基本格式如下：

```
<header>
    <h1>网页主题</h1>
</header>
```

技巧：在一个 HTML 网页中，并不限制 header 元素的个数。一个网页中可以拥有多个 header 元素，也可以为每一个内容块添加 header 元素。

【例 6.1】

在一个完整的网站中，首先会设计网站的页面布局。HTML 5 出现之前，通常会使用以下代码来表示标题：

```
<div id="content">
  <div id="header">
    <div class="left">
      <h1>茶叶模板</h1>
      <h2>各种茶类</h2>
    </div>
  </div>
  <!-- 其他内容 -->
</div>
```

在上述代码中，最外侧的 div 元素搭建整个网站的框架，id 属性值为 header 的 div 元素表示网站的头部信息，该头部信息中包含两个标题，h1 元素指定主标题，h2 元素指定副标题。另外，可以通过不同种类的选择器(例如 ID 选择器和样式选择器)为 div 元素添加 CSS 样式。上述内容的部分 CSS 样式代码如下：

```
#content {
    margin: 0 auto;
    padding: 0;
    width: 770px;
    background: inherit;
    color: #454545;
}
#header {
```

```
    height: 120px;
    background: #000 url(images/top2.jpg) no-repeat top center;
    margin-bottom: 0px;
    color: #454545;
    overflow: hidden;
}
#header .left {
    width: 190px;
    float: left;
    text-align: center;
    padding-left: 14px;
}
```

根据上述描述内容，通过使用 header 元素来替换 id 属性值为 header 的 div 元素。页面相关代码如下：

```
<header>
   <div class="left">
      <h1>茶叶模板</h1>
      <h2>各种茶类</h2>
   </div>
</header>
```

重新更改与 header 元素相关的 CSS 代码，直接通过元素选择器指定样式。也可以说，将使用#header(ID 选择器)设置的代码通过 header(元素选择器)替换。部分代码如下：

```
header {
    height: 120px;
    background: #000 url(images/top2.jpg) no-repeat top center;
    margin-bottom: 0px;
    color: #454545;
    overflow: hidden;
}
```

在浏览器中运行更改后的 HTML 网页，查看效果，如图 6-1 所示。



图 6-1　header元素的使用

6.1.2　article元素

article 元素代表文档、页面或者应用程序中独立的、完整的、可以独自被外部引用的内容。它可以是一篇博客或者报刊中的文章、一篇论坛帖子、一段用户评论或独立的插件，或者其他任何独立的内容。

article 元素可以单独使用，也可以和其他元素结合使用。一个 article 元素通常可以包含它自己的标题，标题一般放在 header 元素中；有时还可以有脚注，脚注一般放在 footer 元素中。

【例 6.2】

下面通过 article 元素显示一首歌曲的内容。其中通过 header 元素嵌入歌曲的标题，歌曲标题放在 h1 元素中，在歌曲的标题下嵌入了一大段该歌曲的歌词，这些内容放在 pre 元素中。代码如下：

```
<article>
    <header>
      <h1>时间都去哪啦</h1>
    </header>
    <pre>
        <p>门前老树长新芽  院里枯木又开花  半生存了好多话  藏进了满头白发</p><p>记忆中的小脚丫  肉嘟嘟的小嘴巴  一生把爱交给他  只为那一声爸妈</p><p>时间都去哪儿了
还没好好感受年轻就老了
生儿养女 一辈子
满脑子都是孩子哭了笑了</p><p>时间都去哪儿了
还没好好看看你眼睛就花了
柴米油盐 半辈子
转眼就只剩下满脸的皱纹了</p>
    </pre>
</article>
```

直接通过元素选择器为 article 元素添加 CSS 样式，内容如下：

```
article {
    width: 736px;
    margin-right: auto;
    margin-left: auto;
    padding: 10px;
    background:  #FFF;
    color: #444;
    display: block;
}
```

在浏览器中运行上述代码查看效果，如图 6-2 所示。



图 6-2　article元素的使用

提示： article 元素是可以进行嵌套的，内层的内容在原则上需要与外层的内容相关联。例如一篇文章，针对该文章的评论就可以使用嵌套 article 元素的方式，用来呈现评论的 article 元素被包含在表示整体内容的 article 元素中。

6.1.3 section元素

section 元素用于对网站或应用程序中页面上的内容进行分块。一个 section 元素通常由内容和标题组成。但是 section 元素并非一个普通的容器元素，当一个容器需要被直接定义样式或通过脚本定义行为时，推荐使用 div 元素，而不是 section 元素。

在使用 section 元素时，需要注意以下 3 点：

- 不要将 section 元素用作设置样式的页面容器，那是 div 元素的工作。
- 如果 article 元素、aside 元素或 nav 元素更符合使用条件，那么不要使用 section 元素。
- 不要为没有标题的内容区块使用 section 元素。

注意： 通常不推荐为那些没有标题的内容使用 section 元素，可以使用 HTML 5 轮廓工具(网址是 http://gsnedders.html5.org/outline/)来检查页面中是否有没标题的 section。如果使用该工具检查后发现某个 section 的说明中有 untitled section(没有标题的 section)文字，这个 section 就有可能使用不当(但是 nav 或 aside 元素没有标题是合理的)。

【例 6.3】

在例 6.2 的基础上添加代码，通常 article 元素的嵌套显示歌曲的评论信息，这些歌词信息作为一个独立的区域进行显示。部分代码如下：

```
<article>
    <!-- 其他内容 -->
    <section>
        <h2>关于这首歌词，有人感动，有人感叹，也有人吐槽。</h2>
        <article>
            <h1>感动篇</h1>
            <p>网友 雨纷飞 说：时间都去哪儿了，看着父母满头白发，心里一点点的愧疚起
来，这么多年，自己为他们做过些什么。时间都去哪了？</p>
            <p>网友 LoveMe 说：爸爸妈妈辛苦了！！</p>
        </article>
        <article>
            <h1>吐槽篇</h1>
            <p>网友 毛毛虫 说：时间都去哪儿了，时间都去这了。(PS：<a href="#">点我
查看</a>)</p>
        </article>
    </section>
</article>
```

为 article 元素下的 h2 元素添加 CSS 样式，样式代码如下：

```
article h2 {
     font-weight: bold;
```

```
    padding: 10px 10px 10px 10px;
    color: #C61C18;
    font-size: 12px;
}
```

在浏览器中运行上述代码，查看效果，如图 6-3 所示。



图 6-3　section元素的使用

在为歌词添加评论信息时，使用到了 section 元素，该元素把歌词正文与评论信息进行了区分。将评论内容进行了分类，每一类都是一段独立的、完整的内容，因此使用多个 article 元素。

section 元素的作用是对页面上的内容进行分块，或者说对文章进行分段。实际上，在 HTML 5 中，article 元素可以看作是一种特殊的 section 元素，它比 section 元素更强调独立性，即 section 元素强调分段或分块，而 article 强调独立性。具体来说，如果一块内容相对来说比较独立、完整时，应该使用 article 元素；但是如果想要将一块内容分成多段时，应该使用 section 元素。

6.1.4 nav元素

nav 元素是一个可以用作页面导航的链接组，其中的导航元素链接到其他页面或当前页面的其他部分。并不是所有的链接组都要被放进 nav 元素，只需要将主要的、基本的链接组放进 nav 元素即可。

一个 HTML 网页中可以包含多个 nav 元素，作为页面整体或者不同部分的导航。具体来说，nav 元素可以用于以下几种场合。

- 传统导航条：目前主流网站上都有不同层级的导航条，其作用是将当前画面跳转到网站的其他主要页面。
- 侧边栏导航：目前主流博客网站及商品网站上都有侧边栏导航，其作用是将页面从当前文章或当前商品跳转到其他文章或其他商品页面。
- 页内导航：它在作用是在本页面几个主要的组成部分之间进行跳转。
- 翻页操作：翻页操作是指在多个页面的前后页或博客网站的前后篇文章滚动。

除了以上几点而外，nav 元素也可以用于其他开发者觉得是重要的、基本的导航链接组中。

【例 6.4】

几乎所有的网站都少不了导航链接，本示例通过 nav 元素定义一个导航链接。实现步骤如下。

步骤01 向 HTML 页面中添加导航链接元素，在该导航元素内嵌入多个超链接元素。代码如下：

```
<nav>
    <a href="#">首页</a> | <a href="#">新闻动态</a> | <a href="#">规范标准
</a> | <a href="#">文章教程</a> | <a href="#">资源链接</a> | <a href="#">常
见问题</a> | <a href="#">论坛交流</a>
</nav>
```

步骤02 为上述 nav 元素和该元素下的 a 元素添加 CSS 样式。以 nav 元素为例，样式代码如下：

```
nav {
    float: left;
    padding-top: 11px !important;
    padding-top: 9px;
    background: transparent no-repeat right top;
    color: #FFF;
    height: 27px;
    text-align: center;
    line-height: 27px;
}
```

步骤03 在浏览器中运行上述代码观察效果，如图 6-4 所示。



图 6-4 nav元素的使用

注意：HTML 5 中不要使用 menu 元素代替 nav 元素，过去有很多 Web 应用程序的开发者喜欢使用 menu 元素进行导航。在这里有必要进行强调：menu 元素是用在一系列发出命令的菜单上的，是一种交互性的元素，或者更确切地说，是使用在 Web 应用程序中的。

6.1.5 aside元素

aside 元素用来表示当前页面或者文章的附属信息部分，它可以包含与当前页面或主要内容相关的引用、侧边栏、广告、导航条，以及其他类似的有别于主要内容的部分。一般

情况下，aside 元素有以下两种用法：

- 被包含在 article 元素中作为主要内容的附属信息，其中的内容可以是与当前文章有关的参考资料、名词解释等。
- 在 article 元素之外使用，作为页面或站点全局的附属信息部分。最典型的形式是侧边栏，其中的内容可以是友情链接、博客中其他文章列表和广告单元等。

【例 6.5】

在介绍文言文的书籍中，会将一些比较复杂、难以理解的字或词放在当前页的底部进行解释。下面通过 aside 元素模拟实现一个类似的功能，效果如图 6-5 所示。



图 6-5 aside元素的效果

步骤01 向 HTML 网页中添加 article 元素，在该元素中嵌入 header 元素显示标题，嵌入 p 元素显示文章内容。代码如下：

```
<article>
    <header>
      <h1>鱼我所欲也</h1>
    </header>
    <p>      鱼，我所欲也，熊掌，亦
<sup>(1)</sup>我所欲<sup>(2)</sup>也，二者不可得兼<sup>(3)</sup>，舍鱼而取熊掌
者也。生，亦我所欲也，义，亦我所欲也，二者不可得兼，舍生而取义者也。生亦我所欲，所欲有
甚于生者，故不为苟得也。死亦我所恶，所恶有甚于死者，故患有所不辟也。如使人之所欲莫甚于
生，则凡可以得生者何不用也。使人之所恶莫甚于死者，则凡可以避患者何不为也！由是则生而有
不用也；由是则可以避患而有不为也。是故所欲有甚于生者，所恶有甚于死者。非独贤者有是心
也，人皆有之，贤者能勿丧耳。<br/>
      <!-- 省略其他内容 -->
    </p>
    <!-- 其他内容 -->
</article>
```

步骤02 在上个步骤的基础上添加代码，向 p 元素之后添加 aside 元素，用来存放名词解释部分的内容：

```
<aside>
  <dl>
    <h1>名词解释</h1>
    <dt>(1)亦：也。</dt>
```

```
      <dt>(2)欲：想要。</dt>
      <dt>(3)兼：同时具有。</dt>
      <dd></dd>
    </dl>
</aside>
```

步骤 03 在浏览器中运行上述代码，查看效果。

6.1.6 footer元素

footer 元素很容易理解，它可以作为其上层父级内容区块或者是一个根区块的脚注。它通常包含相关区块的脚注信息，例如作者、相关阅读链接及版权信息等。在 HTML 5 出现之前，通常都是使用<div id="footer"></div>标记来实现的，HTML 5 出现之后，直接使用 footer 元素来代替。

footer 元素与 header 元素一样，一个页面中可以使用多个 footer 元素。同时，可以为 article 元素或者 section 元素添加 footer 元素。

【例 6.6】

下面的代码在页面的最后使用 footer 元素显示脚注信息：

```
<footer>
    <div id="r">&copy; Copyright 2014, my love website<br />Design: <a 
href="#">Lucy </a></div>
    <div id="l">
        <a href="# ">XHTML</a> - <a href="#">CSS</a>
    </div>
</footer>
```

6.2 分组元素

顾名思义，分组元素就是对页面中的内容进行分组的。HTML 5 中涉及到 3 个与分组有关的元素：hgroup 元素、figcaption 元素和 figure 元素。

6.2.1 hgroup元素

hgroup 元素是将标题及其子标题进行分组的元素，该元素通常会将 h1 到 h6 元素进行分组，例如一个内容区块的标题及其子标题算作一组。

【例 6.7】

观察图 6-1 中的效果和例 6.1 的代码，可以发现，通过 header 元素设计头部时包含了两个标题，一个主标题，一个副标题。其中，主标题通过 h1 元素表示；副标题通过 h2 元素表示。在例 6.1 中通过 div 标记将它们两个进行分组，并且通过 class 设置样式。

HTML 5 中增加 hgroup 元素之后，可以直接通过该元素进行分组。代码如下：

```
<header>
  <hgroup>
    <h1>茶叶模板</h1>
```

```
    <h2>各种茶类</h2>
  </hgroup>
</header>
```

如果有必要，还需要为 hgroup 元素指定 CSS 样式，样式代码和运行效果这里不再进行演示了。hgroup 元素并不是想用就用的，通常情况下，在使用 hgroup 元素时，需要遵循以下几个条件：

- 如果只有一个标题元素(h1~h6 中的一个)，不建议使用 hgroup 元素。
- 当出现一个或者一个以上的标题和元素时，推荐使用 hgroup 元素作为标题容器。
- 当一个标题有副标题、其他 section 或者 article 的元数据时，建议将 hgroup 元素和元数据放到一个单独的 header 元素容器中。

6.2.2 figcaption和figure

在 HTML 5 中，figcaption 元素用于定义 figure 元素的标题，该元素应该被放到 figure 元素的第一个或者最后一个子元素的位置。

在 HTML 5 中，figure 元素指定独立的流内容，例如图像、图表、照片和代码等。

figure 元素的内容应该与主内容无关，如果被删除，则不会对文章流产生影响。在使用 figure 元素时，可以通过 figcaption 元素添加标题。一个 figure 元素内最多只允许放置一个 figcaption 元素，但是允许放置多个其他元素。

【例 6.8】

本例通过 article 显示一篇文章，在文章的最后显示一张图片，并且指定图片的标题。将图片和标题放在 figure 元素中，并且在该元素中嵌入 figcaption 元素，指定特定的标题信息。相关代码如下：

```
<figure>
    <h2>愿得一人心，与你不相离</h2>
    <img src="images/hope.jpeg"/>
</figure>
```

在浏览器中运行上述代码，查看效果，如图 6-6 所示。



图 6-6　figure和figcaption的使用

提示： figure 元素所表示的内容通常是图片、统计图或代码示例，但是并不仅限于此。它同样可以用来表示音频插件、视频插件或者统计表格等。

6.3 文本语义元素

语义元素，即“元素的意义”。具体来说，就是语义元素能够为浏览器和开发者清楚地描述其意义。例如，可以将 header 和 footer 等元素看作是语义元素，而 div 则属于无语义元素。

下面介绍 HTML 5 中新增的几种文本语义元素，它们分别是 mark 元素、ruby、rt 和 rp 元素、time 元素以及 wbr 元素。

6.3.1 mark元素

mark 元素表示页面中需要突出显示或者高亮显示的对于当前用户具有参考作用的一段文字。

mark 元素通常作用在两个方面：第一个方面是对网页全文检索某个关键词时显示的检查结果，目前许多搜索引擎使用其他方法实现该元素所达到的功能；另一个方面是在引用原文时，为了某种特殊目的而把原文作者没有特别重点标示的内容给标示出来。

【例 6.9】

下面的代码显示了一篇文章的部分节选内容，这篇文章的标题是“人生没有假如”，因此在文章的节选部分为了强调“人生没有假如”，通过 mark 元素将这个几个字给高亮显示出来。代码如下：

```
<article>
    <header>
      <h1>人生没有假如</h1>
    </header>
    <p>          面对人生，就要像林黛玉刚进贾府那样就，步步小心，时时在意；不敢多走一步路，不敢多说一句话。人生，得低头时且低头吧，有事，不忍辱负重，又怎可能迎来艳阳天呢！古人曰："唯有埋头，方能出头！"</p>
    <p>      <mark>人生没有假如</mark>，不要把希望放在明天，也不要让人生等待未来，那是很不现实的。明天的日子如何，未来的人生怎样，全在此时此刻的努力。</p>
    <p>      <mark>人生没有假如</mark>，不要对它抱着幻想。忘记你所失去的，珍惜你所拥有的。这样，尽管人生未必没有遗憾，但也可以无悔了。谁都不能对一个努力的生命抱太多的苛责。所以说，不管是怎样的人生，只要尽力了，也就难得了，也就不需要假如了。</p>
</article>
```

在浏览器中运行上述代码，查看效果，如图 6-7 所示。

在 HTML 4 中，还可以使用 em 元素或者 strong 元素突出显示文字，但是 mark 元素的作用是与它们有区别的，不能混同使用。



图 6-7　mark元素的使用

下面分别对 mark、strong 和 em 元素进行说明。

- mark：该元素与原文作者无关，或者说，它不是原文作者用来标示文字的，而是在后来引用时添加上去的，其目的是吸引用户的注意力，提供给用户作为参考，希望对用户有所帮助。
- strong：该元素是原文作者用来强调一段文字的重要性的(例如警告信息)。
- em：该元素是作者为了突出文章重点而使用的。

6.3.2　ruby、rt和rp元素

ruby 定义 ruby 注释，通常与 rt 和 rp 元素一块使用。rt 元素定义 ruby 注释的解释，如果浏览器不支持 ruby 元素显示的内容，就会显示 rp 元素定义的内容。

ruby 注释是中文注音或字符，在东亚使用，显示的是东亚字条的发音。ruby 元素由一个或者多个字符(需要一个解释/发音)和一个提供该信息的 rt 元素组成，还包括可选的 rp 元素。

【例 6.10】

ruby、rt 和 rp 的使用非常简单，下面通过两种方式演示这些元素。代码如下：

```
<ruby>
    漢 <rt><rp>(</rp>ㄏㄢˋ<rp>)</rp></rt>
</ruby>
<ruby>
    汉<rt>ic</rt><rp>五笔拼写：hc</rp>
</ruby>
```

6.3.3　time元素

time 元素定义公历的时间(24 小时制)或日期，时间和时区偏移是可选的。该元素能够以机器可读的方式对日期和时间进行编码。举例来说，用户代理能够把生日提醒或排定的事件添加到用户日程表中，搜索引擎也能够生成更智能的搜索结果。

datetime 和 pubdate 属性是 time 元素常用的两个属性。datetime 属性指定日期/时间，否则，由元素的内容给定日期/时间；pubdate 属性指定 time 元素中的日期/时间是文档(或

article 元素)的发布日期。

【例 6.11】

目前，所有的主流浏览器都不提供对 time 元素的支持。如下代码演示了 time 元素的基本使用：

```
<p>李瑶每天早上<time>7:00</time>起床，每天晚上<time>10:00</time>睡觉。</p>
<p>李瑶<time datetime="2015-02-14">情人节</time>有个约会。</p>
```

6.3.4 wbr元素

wbr 即 Word Break Opportunity，wbr 元素指定在文本中的何处适合添加换行符。如果单词过长，或者开发者担心浏览器会在错误的位置换行，那么也可以使用 wbr 元素来添加单词换行时机。

【例 6.12】

wbr 元素的使用也非常简单，下面的代码演示了该元素的基本使用：

```
<nobr>此行文本不会断行，不管窗口的宽度如何。</nobr>
<nobr>但是，本行，如果<WBR>窗口的宽度太小的话，将在"如果"之后断行。</nobr>
```

6.4 交互元素

HTML 5 是一些独立特性的集合，它不仅增加了许多 Web 页面特征，而且本身也是一个应用程序。对于应用程序而言，表现最为突出的就是交互操作。HTML 5 为操作新增加了对应的交互体验元素，本节来简单了解这些元素。

6.4.1 meter元素

meter 是 HTML5 新追加的用来定义度量衡的元素，该元素仅用于已知最大和最小值的度量。例如，显示硬盘容量或者对某个候选者的投票人数占投票总人数的比例等，都可以使用 meter 元素。

meter 元素的开始标记和结束标记之间可以添加文本，在浏览器不支持该元素时可以显示标记之间的文字。基本格式如下：

```
<meter>浏览器不支持 meter 元素</meter>
```

<meter>标记包含多个属性，如表 6-1 显示了常用的 6 个属性。

表 6-1 meter 元素的常用属性

属性名称	说 明
value	定义需要显示在 min 和 max 之间的值，这是在元素中特地表示出来的实际值。默认值为 0
min	定义允许范围内的最小值，默认值为 0。该属性的值不能小于 0

续表

属性名称	说　明
max	定义允许范围内的最大值，默认值为 1。如果该属性的值小于 min 属性的值，那么把 min 视为最大值
low	定义范围内的下限值，必须小于或等于 high 属性的值。如果该值小于 min，则使用 min 作为 low 属性的值
high	定义范围内的上限值，如果该属性值小于 low，则使用 low 作为 high 的值。如果该值大于 max，则使用 max 作为 high 属性的值
optimum	最佳值，其值必须在 min 属性值与 max 属性值之间，可以大于 high 属性值

【例 6.13】

某实验中学一班在开学时进行班长选举活动，该班级有 50 名同学，每位同学最多选择两票，徐海、陈露和李雪三名同学参加了这次选举活动。下面的代码通过 meter 元素表示这 3 名同学的支持率：

```
<p>
    徐海：<meter low="69" high="80" max="100" optimum="100" value="92">A</meter>
    陈露：<meter low="69" high="80" max="100" optimum="100" value="72">C</meter>
    李雪：<meter low="69" high="80" max="100" optimum="100" value="52">E</meter>
</p>
```

运行上述代码观察效果，如图 6-8 所示。

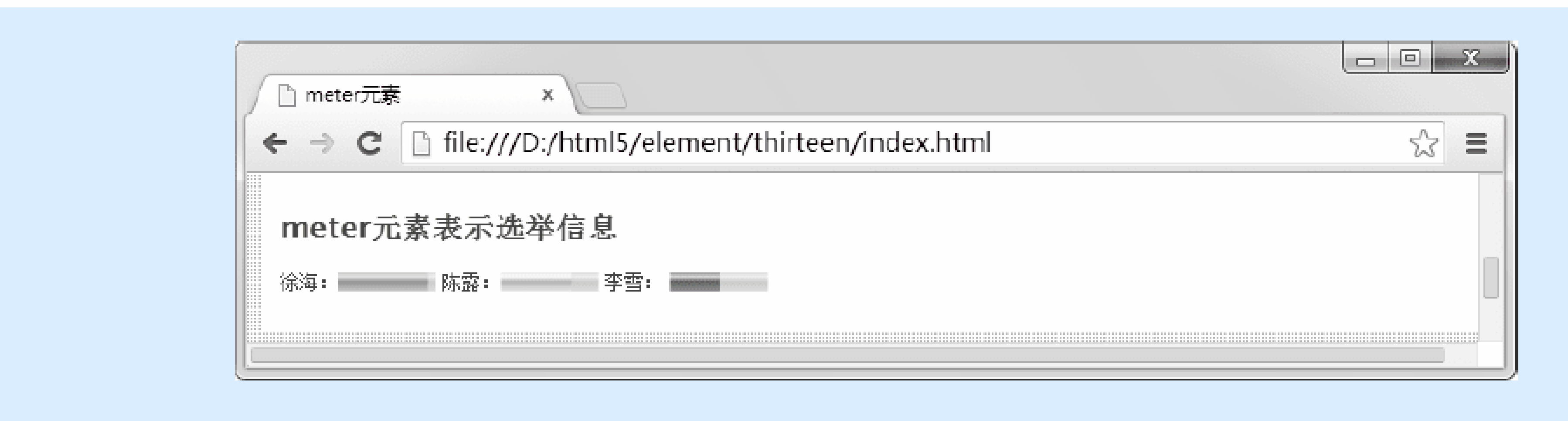


图 6-8　使用meter元素的初始效果

图 6-8 显示了 meter 元素的默认样式，如果不使用该元素的默认样式，开发者也可以自定义样式。例如，下面通过样式选择器指定 CSS 样式，如下代码仅适用于 WebKit 内核的浏览器：

```
.deal meter { -webkit-appearance: none; }
.deal ::-webkit-meter-bar {
    height: 1em;
    background: white;
    border: 1px solid black;
}
.deal ::-webkit-meter-optimum-value { background: green; }      /*好*/
.deal ::-webkit-meter-suboptimum-value { background: orange; }  /*凑合*/
.deal ::-webkit-meter-even-less-good-value { background: blue; } /*糟糕*/
.deal ::-moz-meter-bar {
    background: rgba(0,96,0,.6);
}
```

为显示支持信息的<p>标记添加 class 属性，代码如下：

```
<p class="deal">
  <!-- 省略其他内容 -->
</p>
```

重新运行页面或者刷新浏览器页面，自定义效果如图 6-9 所示。



图 6-9 自定义meter元素样式

6.4.2 progress元素

progress 直译过来是“发展、进步、进度”的意思，顾名思义，progress 元素代表一个任务的完整进度，这个进度可以是不确定的，只是表示进度正在进行，但是不清楚还有多少工作量没有完成。可以使用 0 到某个最大数(例如 100)之间的数值来表示进度完成的准确情况。progress 元素具有两个属性，来表示当前任务完成的情况：value 属性表示已经完成了多少工作量；max 属性表示总共有多少工作量，工作量单位是随意的，不用指定。在设置属性时，value 属性和 max 属性只能指定为有效的浮点数，value 属性的值必须大于 0，且小于或等于 max 属性，max 属性的值必须大于 0。

【例 6.14】

可以使用 progress 元素显示 JavaScript 中耗费时间的函数的进度。本示例通过两个 progress 元素进行演示，第一个 progress 元素设置 value 和 max 属性的值，第二个 progress 元素只设置 max 的值，并且只在单击按钮时控制进度条。实现步骤如下。

步骤01 向 HTML 页面中添加第一个 progress 元素，指定该元素的 max 属性和 value 属性。代码如下：

```
<section>
    <h1>progress 元素的使用 2</h1>
    <p><progress value="45" max="100"><span>45%</span></progress></p>
</section>
```

步骤02 向 HTML 页面中添加第二个 progress 元素，指定该元素的 max 属性值，然后添加一个按钮，并为该按钮添加 onClick 事件属性。代码如下：

```
<section>
    <h1>progress 元素的使用 2</h1>
    <p>完成百分比：
      <progress id="p" max="100"><span>0</span>%</progress></p>
    <input type="button" onClick="button click()" value="请点击" />
</section>
```

步骤03 向 JavaScript 脚本中添加 button_click()函数，在该函数中定义 progress 元素的值。当用户单击按钮时，progress 元素就会自动增长，当它的值增长到 100(即 progress 元素的 max 值)时，就会停止增长。JavaScript 脚本代码如下：

```
<script type="text/javascript">
    var newValue = 0;
    function button_click(){
        //获取页面中的 progress 元素
        var progressBar = document.getElementById('p');
        newValue = 0;        //设置 newValue
        progressBar.getElementsByTagName('span')[0].textContent = 0;
        setTimeout("updateProgress()",500);
    }
    function updateProgress(){
        if(newValue>100){
            return ;
        }
        var progressBar = document.getElementById('p');
        progressBar.value = newValue;
        progressBar.getElementsByTagName('span')[0].textContent=newValue;
        setTimeout("updateProgress()",500);
        newValue++;
    }
</script>
```

步骤04 在浏览器中运行上述代码，查看效果，图 6-10 和 6-11 分别显示了 Chrome 浏览器和 Firefox 浏览器的初始效果。

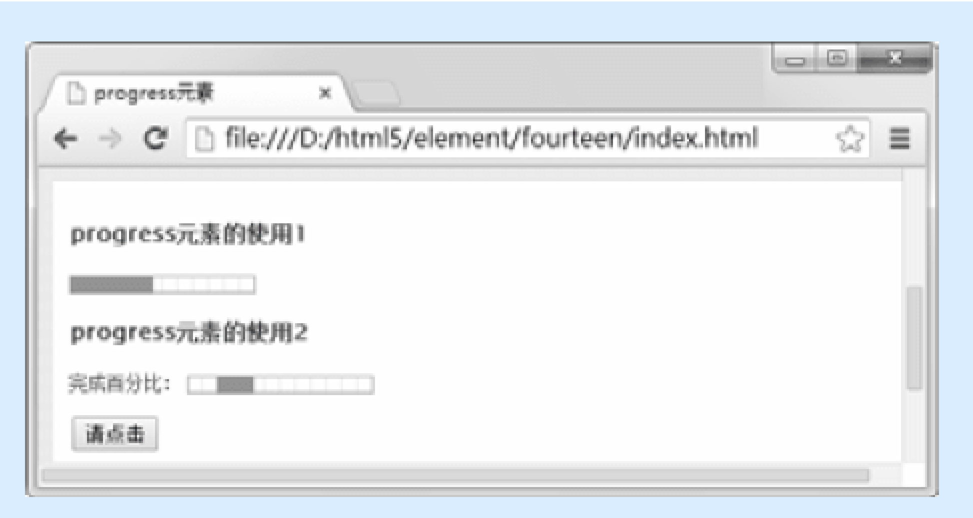


图 6-10　Chrome浏览器的初始效果

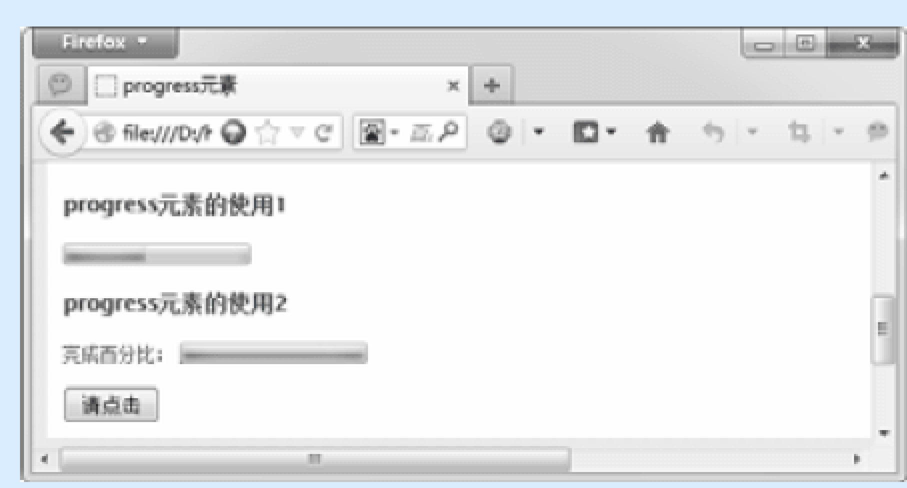


图 6-11　Firefox浏览器的初始效果

步骤05 单击图中的“请点击”按钮查看效果，图 6-12 和 6-13 分别为 Chrome 浏览器和 Firefox 浏览器的效果。



图 6-12　在Chrome浏览器中单击按钮

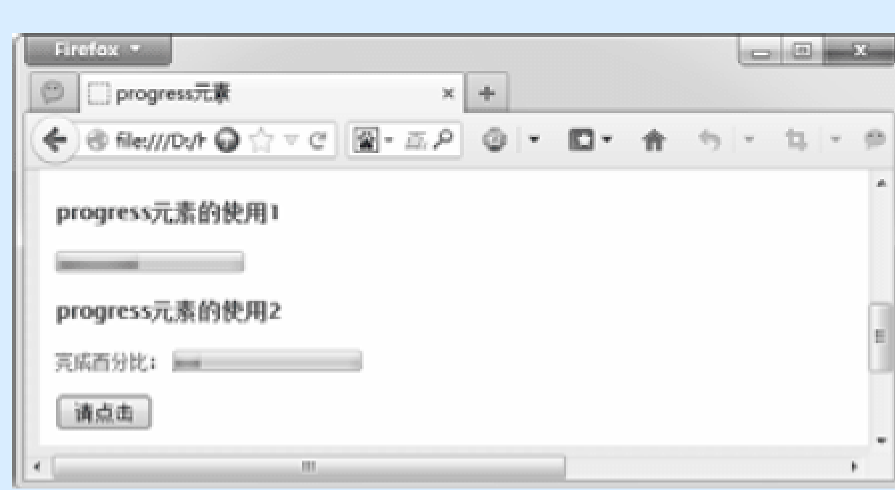


图 6-13　在Firefox浏览器中单击按钮

6.4.3 details元素

details 元素提供了一种替代 JavaScript 的、将画面上局部区域进行展开或收缩的方法，用于说明文档或某个细节信息的作用。<details>标记经常会使用到一个 open 属性，该属性定义 details 是否可见。

【例 6.15】

本例向页面中分别添加两个 details 元素，并为第二个 details 元素指定 open 属性。代码如下：

```
<details>
          东方明珠广播电视塔(the Oriental
Pearl TV Tower)，坐落在中国上海浦东新区陆家嘴，毗邻黄浦江，与外滩隔江相望，上海国际
新闻中心所在地。东方明珠塔是由上海现代建筑设计(集团)有限公司的江欢成设计。塔高 468 米。
</details>
<details open="open">
          埃菲尔铁塔(法语: La Tour Eiffel)是一
座于 1889 年建成位于法国巴黎战神广场上的镂空结构铁塔，高 300 米，天线高 24 米，总高 324
米。埃菲尔铁塔得名于设计它的著名建筑师、结构工程师居斯塔夫·埃菲尔。铁塔设计新颖独特，
是世界建筑史上的技术杰作，是法国巴黎的重要景点和突出标志。
</details>
```

在浏览器中运行上述代码，观察效果，如图 6-14 所示。从图中可以看出，为<details>标记指定 open 属性后，页面在运行时会自动展开显示内容。

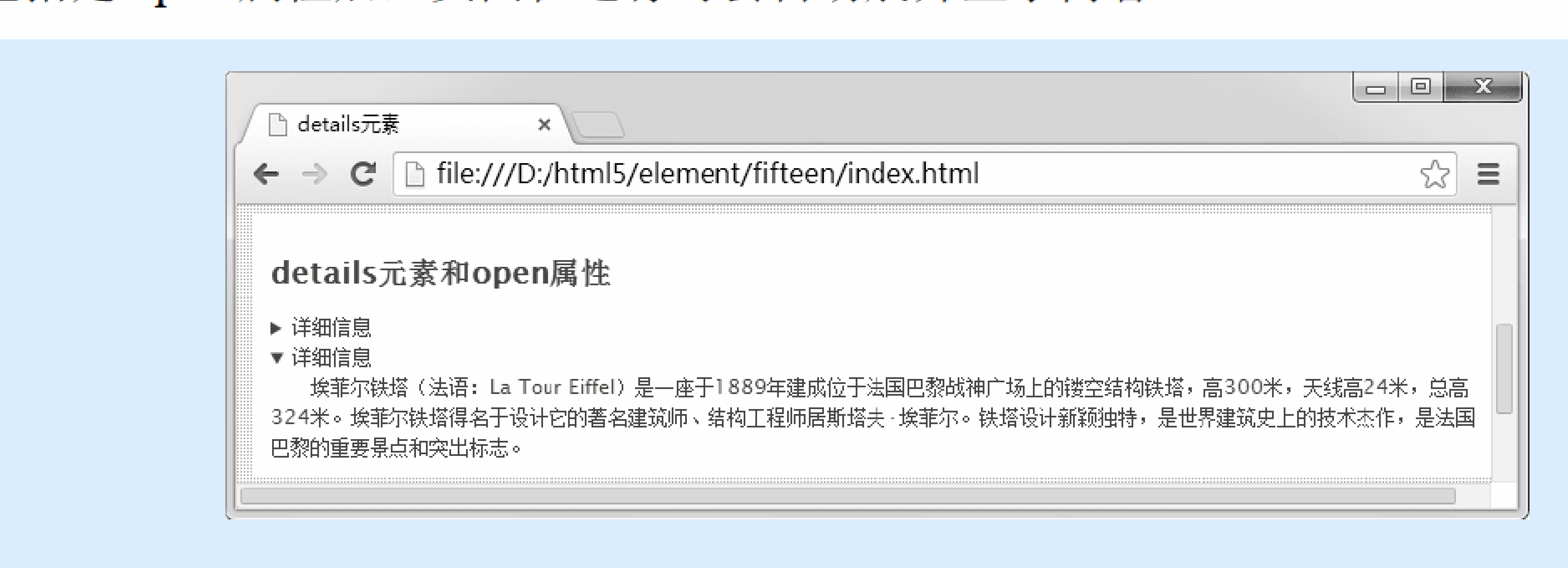


图 6-14 details元素的使用

6.4.4 summary元素

观察上个示例的代码和效果图，可以看出，details 元素只添加了显示的内容，并没有为其指定标题，这时浏览器会提供一个默认的标题，即“详细信息”，同时提供一个诸如上下箭头之类的图标，单击该图标可以进行展开和收缩操作。如果要为 detail 元素自定义标题，这时可以使用 summary 元素，该元素用来定义 details 元素的标题。

【例 6.16】

重新更改上个示例的代码，为每一个 details 元素指定标题。以第一个 details 元素为例，部分代码如下：

```
<details>
```

```
    <summary>东方明珠广播电视塔</summary>
    <!-- 省略其他内容 -->
</details>
```

在浏览器中运行例 6.16 的代码，查看效果，并且单击图中的箭头图标进行展开和收缩操作，如图 6-15 所示。

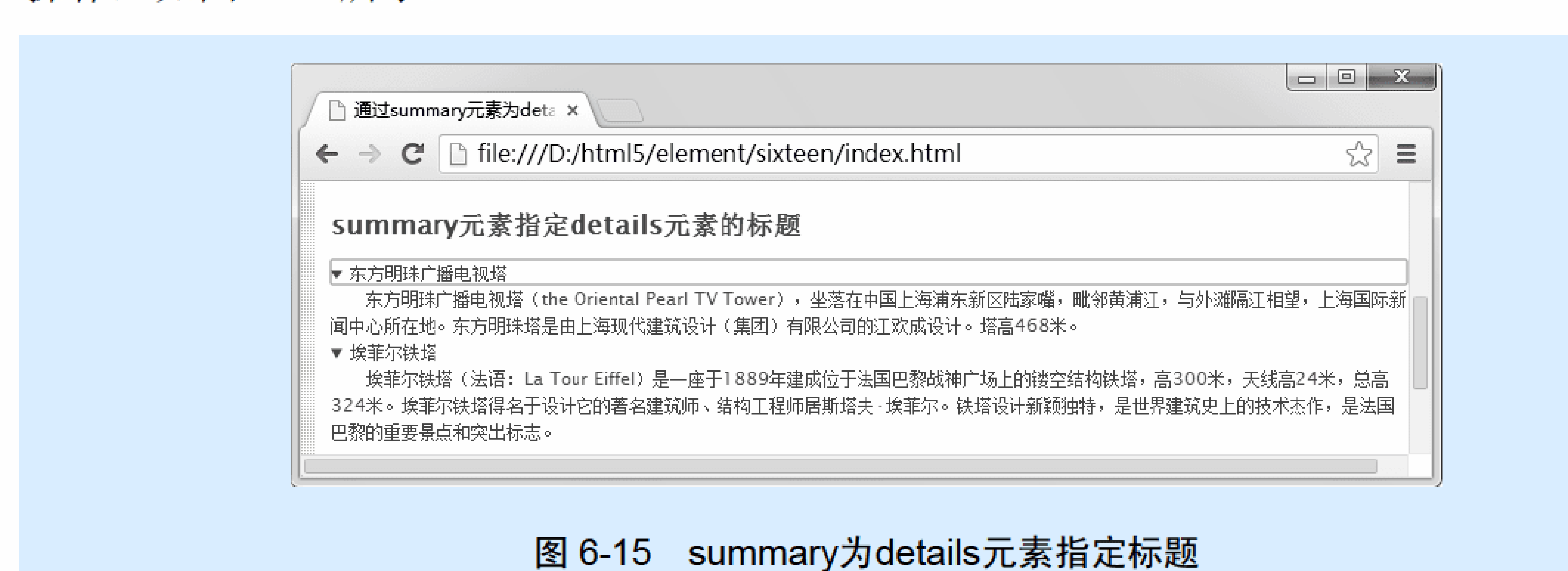


图 6-15　summary为details元素指定标题

注意：HTML 5 中新增了多个交互元素，除了本节介绍的几个元素外，command 元素和 dialog 元素都可以看作是交互元素。但是，目前，还没有浏览器提供对它们的支持操作，因此这里不再进行详细的介绍。

6.5　音频和视频元素

作为最重要的 Web 开发标准的下一代，HTML 5 引起了很多 Web 开发者的关注。它最重要的一个特性就是对音频和视频的支持，例如构建音频可视化，在线视频编辑等，这一突破为互联网多媒体技术带来更多的可发展空间，为多媒体技术的可协同编辑提供了更好的平台。

本节分别介绍 HTML 5 中新增的 video 元素和 audio 元素，包括这两个元素的属性，以及视频和音频的显示操作等内容。

6.5.1　video元素

video 元素专门用来播放网络上的视频或电影，通过使用该元素，可以把指定的视频数据直接嵌入在网页中。video 元素支持 3 种常用的视频格式：Ogg、MPEG 4 和 WebM。

- Ogg：带有 Theora 视频编码和 Vorbis 音频编码的 Ogg 文件。
- MPEG 4：带有 H.264 视频编码和 AAC 音频编码的 MPEG 4 文件。
- WebM：带有 VP8 视频编码和 Vorbis 音频编码的 WebM 文件。

1. video元素的常用属性

目前，支持 HTML 5 的主流浏览器对视频格式的支持有所不同。video 元素的基本使

用形式如下：

```
<video src="URL" width="宽度" height="高度" controls autoplay preload loop
poster="URL">
    浏览器不支持 video 元素
</video>
```

从上述形式中可以看出，video 元素包含 src、width、height 和 controls 等多个属性，下面对这些属性进行了说明。

- src：指定媒体(这里指视频)数据的 URL 地址。
- width：video 元素的特有属性，指定视频的宽度。
- height：video 元素的特有属性，指定视频的高度。
- controls：指定是否为视频添加浏览器自带的播放用的控制条，控制条中具有播放、暂停等按钮。
- autoplay：指定媒体(这里指视频)是否在页面加载后自动播放，指定该属性时表示自动播放。
- preload：指定视频是否预加载。如果使用预加载的话，浏览器会预先将视频数据进行缓冲，这样可以加快播放的速度，因为播放时数据已经预先缓冲完毕。preload 属性包含 3 个可选择的值，说明如下。
    - none：表示不进行预加载。
    - metadata：表示只预加载媒体的元数据(媒体字节数、第一帧、播放列表和持续时间等)。
    - auto：默认值，表示预加载全部视频。
- loop：指定是否循环播放视频。
- poster：video 元素的特有属性，当视频不可用时，可以使用该属性向用户展示一幅替代用的图片。当视频不可用时，最好使用该属性，以免展示视频的区域中出现一片空白。

注意：无论是本节介绍的 video 元素，还是下一节介绍的 audio 元素，它们的属性不止本节介绍的这些，但其他属性没有这些属性使用频繁，因此这里不再对它们多做介绍了。

【例 6.17】

本示例通过一个详细的步骤向读者演示<video>标记及其属性的使用。实现步骤如下。

步骤01 向 HTML 网页中添加<video>标记，并为该标记指定 controls 属性和 src 属性。页面代码如下：

```
<video controls src="http://www.w3school.com.cn/i/movie.ogg">
    很抱歉，您当前使用的浏览器不支持 video 元素
</video>
```

步骤02 在浏览器中运行上述代码，不同的浏览器显示的效果可能有所不同，图 6-16 和 6-17 分别显示了 Chrome 浏览器和 Firefox 浏览器中的效果。



图 6-16　Chrome浏览器中的效果

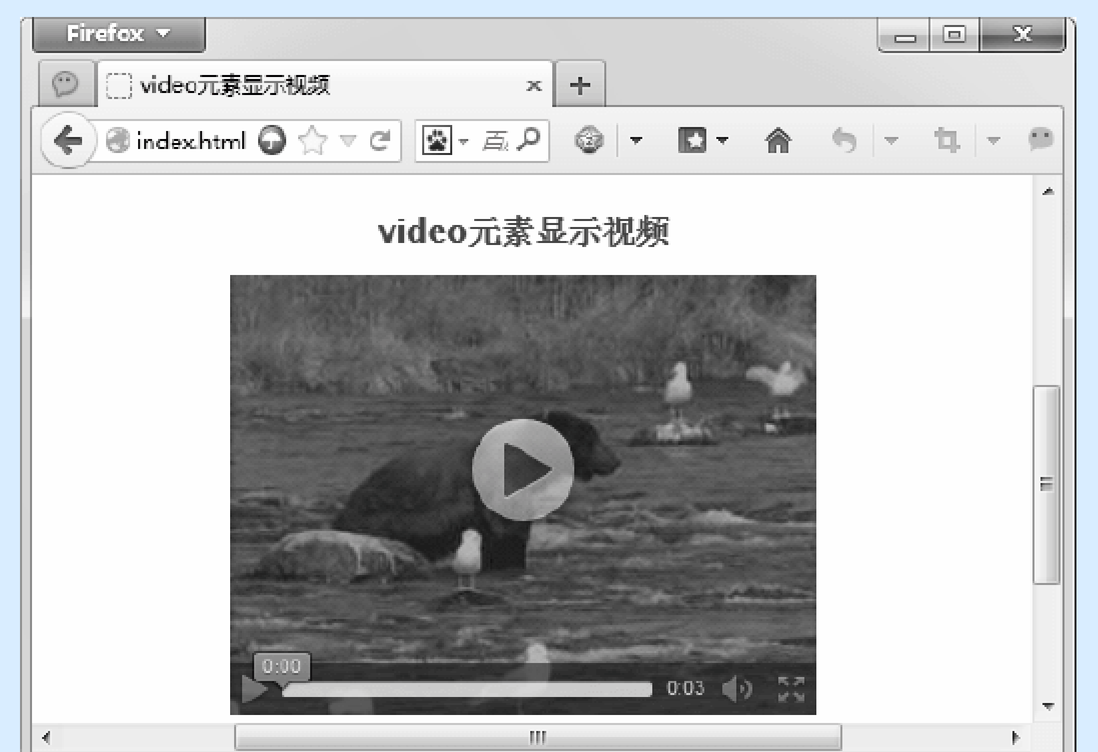


图 6-17　Firefox浏览器中的效果

步骤03 继续向上述网页中添加<video>元素的属性，分别通过 width 和 height 设置视频的宽度和高度。代码如下：

```
<video controls src="http://www.w3school.com.cn/i/movie.ogg" width="300"
height="200">
    很抱歉，您当前使用的浏览器不支持 video 元素
</video>
```

步骤04 重新运行上述代码进行测试，图 6-18 和 6-19 分别为 Chrome 浏览器和 Firefox 浏览器中的效果。



图 6-18　Chrome浏览器中的效果



图 6-19　Firefox浏览器中的效果

步骤05 当视频不可用时，可以使用 poster 属性设置一张图片来代替视频。为了演示 poster 属性的效果，更改本示例的代码，指定播放一个 mp4 文件。代码如下：

```
<video controls src="../filelist/video.mp4" width="300" height="200"
poster="../filelist/error.jpg">
    很抱歉，您当前使用的浏览器不支持 video 元素
</video>
```

步骤06 重新在浏览器中运行上述代码，Chrome 浏览器和 Firefox 浏览器中的效果如图 6-20 和 6-21 所示。



图 6-20 Chrome浏览器中的效果

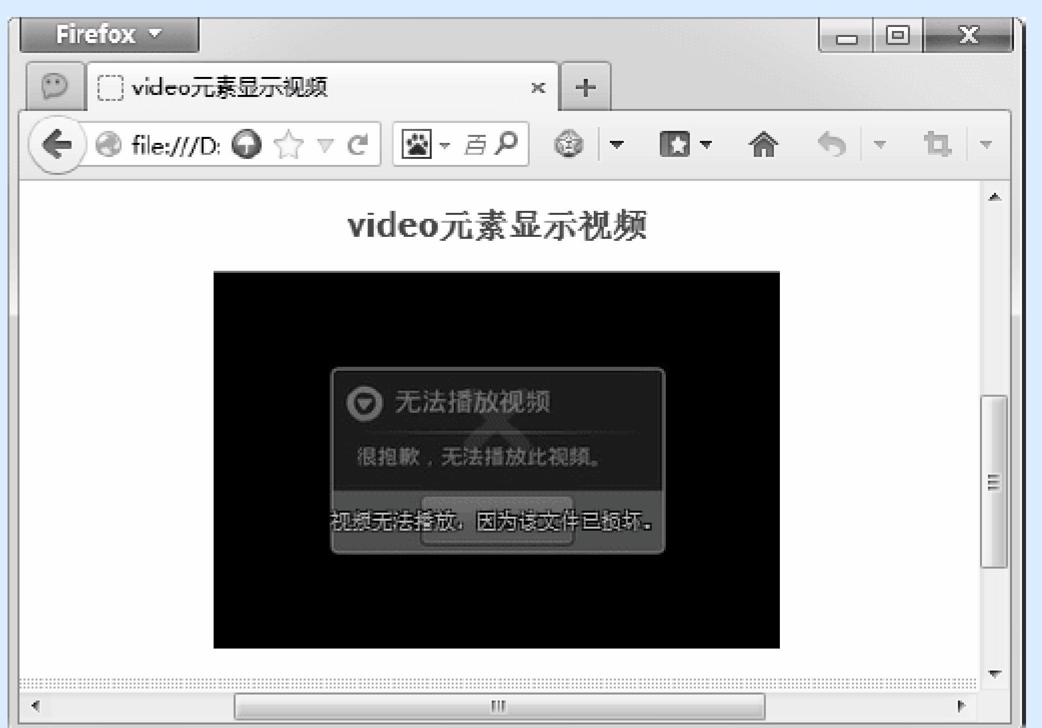


图 6-21 Firefox浏览器中的效果

观察图 6-20 和 6-21 可以发现，Firefox 浏览器不支持 MP4 格式的视频文件，虽然 Chrome 浏览器中也显示了 poster 属性设置的图片，但是实际上它已经加载到了视频文件，0:29 表示视频文件的总播放时间，单击播放按钮时可以自动播放。

2. video元素的方法和事件

video 元素除了包含属性外，还包含一系列的方法和事件，在上一章介绍 HTML 5 的新增属性时，已经介绍了与多媒体元素有关的事件属性，因此，这里不再对 video 或者 audio 元素的事件进行详细介绍，如果有需要，可以参考第 5 章的表 5-9。

下面列出了 video 元素的 3 种常用方法。

- play()：使用 play()方法来播放媒体，自动将元素的 paused 属性的值变为 false。
- pause()：使用 pause()方法来暂停播放，自动将元素的 pause 属性的值变为 true。
- load()：使用 load()方法来重新载入媒体进行播放，自动将元素的 playbackRate 属性值变为 defaultPlaybackRate 属性的值，自动将元素的 error 的值变为 null。

【例 6.18】

本例中，用户通过单击网页中的按钮，来控制视频的播放和暂停功能，播放完毕后弹出提示。实现步骤如下。

步骤01 向 HTML 网页中添加<video>标记，并且指定该标记的 id 属性、src 属性、width 属性和 height 属性。代码如下：

```
<video id="video1" src="http://www.w3school.com.cn/i/movie.ogg"
width="300" height="200">
    很抱歉，您当前使用的浏览器不支持 video 元素
</video>
```

步骤02 在 video 元素之后添加两个 button 元素，分别控制视频的播放和暂停功能，并为该按钮添加 onClick 事件属性。代码如下：

```
<button onClick="play()">播放</button>
<button onClick="pause()">暂停</button>
```

步骤03 通过 JavaScript 脚本分别编写 init()、play()和 pause()函数，其中 init()函数执行初始化操作，play()函数执行播放操作，pause()函数执行暂停播放操作。函数的

完整代码如下：

```
<script>
    var video;
    function init(){
        video = document.getElementById("video1");
        //监听视频播放结束事件
        video.addEventListener("ended",function(){
            alert("播放结束");
        },true);
        //发生错误
        video.addEventListener("error",function(){
            switch(video.error.code){
                case MediaError.MEDIA ERROR ABORTED:
                    alert("视频的下载过程被中止");
                    break;
                case MediaError.MEDIA ERROR NETWORK:
                    alert("网络发生故障，视频的下载过程被中止");
                    break;
                case MediaError.MEDIA ERROR DECODE:
                    alert("解码失败");
                    break;

                case MediaError.MEDIA ERROR SRC NOT SUPPORTED:
                    alert("不支持播放的视频格式");
                    break;
                default:
                    alert("发生未知错误");
            }
        },false);
    }
    function play(){
        video.play();       //播放视频
    }
    function pause(){
        video.pause();      //暂停播放
    }
</script>
```

在 init()函数中，首先获取网页中的 video 元素，然后为该元素添加两个 ended 事件和 error 事件，在视频播放完毕和发生错误时会弹出提示信息。在 play()函数中，调用 play()方法播放视频。在 pause()函数中，调用 pause()方法暂停播放。

步骤 04 为页面中的 body 元素添加 onload 事件，该事件调用 init()函数。

步骤 05 在浏览器中运行本示例，查看效果，如图 6-22 显示了播放完毕后的效果。

3. 实现视频的兼容

不同的浏览器支持的视频格式可能不同，例如 Firefox 浏览器不支持 MP4 格式的视频，但是它支持其他格式(例如 Ogg)的视频。如果要实现浏览器的兼容效果，可以向 video 元素中添加多个 source 元素，source 元素可以链接不同的视频文件，浏览器按 source 元素的顺序检测指定的视频能否播放，如果不能，则播放下一个。



图 6-22　视频播放结束时的效果

<source>标记常用的 3 个属性及其说明如下。

- src：用于指定媒体的地址，与 video 元素和 audio 元素的 src 一样。
- type：用于说明 src 属性指定媒体文件的类型，帮助播放器判断要播放的媒体内容的类型。
- media：用于说明媒体在何种介质中使用，不设置时默认值为 all，表示支持所有介质。

【例 6.19】

向 video 元素中嵌入多个 source 元素，实现浏览器的兼容效果。代码如下：

```
<video width="300" height="200" controls autoplay>
   <source src="../filelist/test.mp4" type="video/mp4" />
   <source src="../filelist/test.webm" type="video/webm" />
   <source src="../filelist/test.ogg" type="video/ogg" />
</video>
```

重新在浏览器中运行上述代码，查看效果。

6.5.2 audio元素

audio 元素专门用来播放网络上的音频数据。audio 元素的大多数属性都与 video 元素相同，但是需要注意的是：audio 元素不能通过 width 和 height 属性设置音频播放器的宽度和高度，如果要设置，需要通过 CSS 样式指定。另外，audio 元素也没有 poster 属性，不能通过指定该属性显示一张图。

在通过 CSS 样式设置播放器的宽度和高度时，宽度可以任意进行调整，但是高度最好不要小于 45px。如果高度大于 45px 时，并不会改变播放控件的高度，但它会在播放控件上增加空白单位；一旦小于 45px，那么就会只显示播放和暂停控件。

【例 6.20】

本例通过一个详细的步骤向读者演示<audio>标记及其属性的使用。

步骤 01　向 HTML 网页中添加<audio>标记，并且为该标记指定 src 属性和 controls 属

性。代码如下：

```
<audio src="../filelist/wheretime.mp3" controls>
    当前浏览器不支持 audio 元素
</audio>
```

步骤 02 分别在 Chrome 浏览器和 Firefox 浏览器中运行上述代码，其效果如图 6-23 和 6-24 所示。



图 6-23　Chrome浏览器中的效果



图 6-24　Firefox浏览器中的效果

步骤 03 分别单击图 6-23 和 6-24 中的播放按钮，来播放音频文件，播放时的效果如图 6-25 和 6-26 所示。



图 6-25　Chrome浏览器中播放文件



图 6-26　Firefox浏览器中播放文件

步骤 04 向<audio>标记添加 autoplay 属性、loop 属性和 preload 属性，然后运行浏览器进行测试。

步骤 05 通过 CSS 样式设置<audio>标记的宽度和高度，然后运行浏览器进行测试。

audio 元素与 video 元素一样，可以通过向 audio 元素中嵌入多个 source 元素的方式实现浏览器的兼容效果。

【例 6.21】

向 audio 元素中嵌入两个 source 元素，这两个 source 元素分别播放.mp3 和.ogg 格式的音频文件，并且需要为播放器定义统一的宽度和高度。代码如下：

```
<audio controls style="width:400px; height:45px;">
    <source src="../filelist/wheretime.mp3" type="audio/mpeg" />
    <source src="../filelist/wheretime.ogg" type="audio/ogg" />
</audio>
```

注意：audio 元素的绝大多数属性都与 video 元素一样，不同的属性本节开始已经介绍过。另外，audio 元素也包含一系列的方法和事件，这些方法和事件都与 video 元素一致。读者可以在上述示例的基础上添加新的代码，练习使用 audio 元素的方法和事件。

6.6 标准属性

上一章在介绍 HTML 5 中的新增标准属性时，提到过多个标准属性，这些标准属性可以对任何元素使用(极特别的元素除外)，本节向读者介绍一些常用的标准属性。

6.6.1 hidden属性

hidden 是 HTML 5 新增的一个属性，该属性类似于 hidden 类型的 input 元素，功能是通知浏览器不显示该元素，使元素处于不可见状态。但是元素中的内容还是浏览器创建的，也就是页面装载后允许使用 JavaScript 脚本将该属性取消，取消后该元素变为可见状态，同时元素中的内容也即时显示出来。hidden 属性是一个布尔值的属性，当将属性值设置为 true 时，元素处于不可见状态；当属性值设置为 false 时，元素处于可见状态。

【例 6.22】

本示例通过 p 元素显示一段文本，并且通过 hidden 属性将这段文本进行隐藏，当用户单击页面中的按钮时可以再次显示这段内容。实现步骤如下。

步骤01 向 HTML 网页的合适位置添加<p>标记，并指定该标记的 hidden 属性。部分代码如下：

```
<input type="button" id="btn" value="显示" onClick="show()" />
<p hidden id="showp">
          Happiness is not about being
immortal nor having food or rights in one's hand. It's about having each
tiny wish come true, or having something to eat when you are hungry or
having someone's love when you need love.
</p>
```

步骤02 添加一段 JavaScript 脚本，脚本内容用于显示或者隐藏 p 元素中指定的内容。代码如下：

```
<script>
var btn = document.getElementById("btn");
function show(){
    var p = document.getElementById("showp");
    if(btn.value == "显示"){
        btn.value = "隐藏";
        p.hidden = false;
    }else{
        btn.value = "显示";
        p.hidden = true;
    }
}
```

```
</script>
```

步骤 03 在浏览器中运行本示例的代码进行测试，图 6-27 为显示内容时的效果。



图 6-27　hidden属性的使用

6.6.2　contenteditable属性

contenteditable 属性的功能是允许用户编辑元素中的内容，因此需要注意，元素必须是可以获得鼠标焦点的元素，而且在点击鼠标后要向用户提供一个插入符号，提示用户该元素中的内容允许编辑。contenteditable 属性与 hidden 属性一样，它也是一个布尔值，可以将该属性的值设置为 true 或 false。contenteditable 属性有一个隐藏的 inherit(继承)状态，属性值为 true 时，元素被指定为允许编辑；属性值为 false 时，元素被指定为不允许编辑；如果没有指定 true 或 false，则由 inherit 状态来决定，如果元素的父元素是可编辑的，则该元素就是可编辑的，否则为不可编辑。

另外，除了 contenteditable 属性外，元素还具有一个 iscontenteditable 属性，当元素可编辑时，属性值为 true，当元素不可编辑时，属性值为 false。

【例 6.23】

下面通过<p>标记显示一段文本，并且将该标记 contenteditable 属性的值设置为 true。代码如下：

```
<p contenteditable="true">
          Happiness is not about being
immortal nor having food or rights in one's hand. It's about having each
tiny wish come true, or having something to eat when you are hungry or
having someone's love when you need love. </p>
```

在浏览器中运行上述代码，在网页中点击这段文字并进行编辑，图 6-28 和 6-29 分别显示了 Chrome 浏览器和 Opera 浏览器的效果。



图 6-28　Chrome浏览器的编辑效果



图 6-29　Opera浏览器的编辑效果

6.6.3　spellcheck属性

spellcheck 属性是 HTML 5 针对 input 元素(type=text)与 textarea 这两个文本输入框提供的一个新属性，其功能是对用户输入的文本内容进行拼写和语法检查。spellcheck 属性的值是一个布尔类型，它在书写时必须将属性值设置为 true 或者 false。

注意：如果为元素指定 readOnly 属性或者 disabled 属性，并且将属性值设置为 true 时，即使设置了 spellcheck 属性也不会执行拼写检查。

【例 6.24】

本示例演示 spellcheck 属性的使用，为了演示该属性值的效果，向页面中添加两个 textarea 元素，分别指定 spellcheck 属性的值为 true 和 false。代码如下：

```
spellcheck 属性值为 true：<br/><textarea row="10" cols="100"
  spellcheck="true"></textarea><br/>
spellcheck 属性值为 false：<br/><textarea row="10" cols="100"
  spellcheck="false"></textarea>
```

在浏览器中运行本示例的代码，并输入内容进行测试，图 6-30 为 Opera 浏览器的测试效果。



图 6-30　spellcheck属性的使用

6.7　实战——使用 HTML 5 元素构建网页

在本节之前，已经介绍过 HTML 5 中新增的结构元素、分组元素、文本语义元素、交互元素、音频和视频元素，以及常用的标准属性等多个内容。本节实战将前面的知识点结合起来构建一个网页，最终的构建效果如图 6-31 所示。



图 6-31　实战运行效果

从图 6-31 中可以看到，本次实战构建的网页非常简单，在添加内容之前，首先分析网页结构。页面最外层通过一个 div 元素进行控制，它包含 4 部分：头部信息、导航链接、文章节选列表以及底部信息。

根据上述的部分内容添加代码进行实现，完整的实现步骤如下所示。

步骤 01　页面头部的内容很简单，只包含一张背景图和 5 个超链接。页面代码如下：

```
<header class="clearfix">
    <div id="mylogo"><img src="img/logo.jpg" alt="村">
      <h1>唐小轩的爱心小屋</h1>
    </div>
    <div id="iconNav" class="clearfix">
      <ul class="clearfix">
        <li class="about"><a href="#about">关于我</a></li>
        <li class="emailme"><a href="#contact">联系我</a></li>
```

```
        <li class="goodies"><a href="#goodies">文章列表</a></li>
        <li class="links"><a href="#links">收藏内容</a></li>
        <li class="awards"><a href="#awards">相册</a></li>
      </ul>
    </div>
  </header>
```

步骤02 为上述代码中的 header、div 和 ul、li 等元素添加 CSS 样式，这里只给出 header 元素的样式。代码如下：

```
header {
    position: relative;
    background-color: #422f2c;
    height: 130px;
    background-image: url(../img/headerBg.gif);
    background-repeat: repeat-x;
    color: #fff;
}
```

步骤03 添加一系列的导航链接，导航链接的外侧通过 nav 元素进行控制，在该元素中再嵌入一个无序列表。页面代码如下：

```
<nav class="clearfix">
    <ul class="clearfix">
      <li><a href="#">首页</a></li>
      <li><a href="#">文章精选</a></li>
      <li><a href="#">相册列表</a></li>
      <li><a href="#">内容收藏</a></li>
      <li><a href="#">心情列表</a></li>
      <li><a href="#">联系我吧</a></li>
    </ul>
</nav>
```

步骤04 为上个步骤中的 nav 元素添加 CSS 样式，代码如下：

```
nav {
    background-image: url(../img/navBg.gif);
    background-repeat: repeat;
    background-color: #f4f9f8;
}
```

步骤05 继续添加与中间内容有关的代码，中间部分通过 article 元素进行控制，然后在该元素中包含多个 section 元素，每一个 section 元素又包含标题和内容等。基本结构如下：

```
<article class="clearfix">
    <div id="web" class="clearfix">
        <h1>精选文章</h1>
        <section></section>
        <section></section>
    </div>
</article>
```

步骤06 为上步中的 article 元素添加 CSS 样式，指定 padding 属性的值。代码如下：

```
article {
    padding: 0 25px;
}
```

步骤07 以第一个 section 为例进行介绍，向该元素中嵌入 header 元素，它包含主标题和副标题两部分，因此，将两个标题放到 hgroup 元素中。代码如下：

```
<header class="test">
   <hgroup class="test1">
      <h1>我的青春我做主(节选)</h1>
      <h2>看《我的青春谁做主》的感悟</h2>
   </hgroup>
</header>
```

步骤08 重新为上个步骤中 header 元素、hgroup 元素和 h1 元素等设置 CSS 样式。以 h1 元素为例，相关代码如下：

```
hgroup h1{
    background-color: white;
    background-image: none;
    position: inherit;
    width: auto;
    height: auto;
    left: 0px;
    padding: 0px 0px 0px 0px;
    color: #000;
}
```

步骤09 继续在上个步骤的基础上添加代码，使用 p 元素显示文章节选的段落内容，如果内容过多，则通过 details 元素表示，将剩下的内容收起来。以 Details 元素为例，代码如下：

```
<details>
    <summary>读取更多</summary>
    <p>     赵青楚，那么的理智，那么的头脑清晰，令人羡慕啊。我自认为很理智但是不够头脑清晰，往往犯糊涂，但是大致方向是理智的。人不可以太理智，该感情用事的时候也要感性一些，否则自己会后悔的。喜欢他对爱情的态度，那么清楚又那么投入。我知道现实生活中可能没有这样完美的人，但是，电视剧源于生活却高于生活。虽然很难做到但是我们要那个方向去努力。</p>
</details>
```

步骤10 通过 figure 元素显示一段独立的视频文件，并且通过 figcaption 元素指定标题。显示视频使用 video 元素，并向该元素中嵌入多个 source 元素。代码如下：

```
<figure>
    <figcaption>第一集视频第一部分观看</figcaption>
    <video controls width="300" height="200" loop>
        <source src="01.mp4" type="video/mp4" />
        <source src="01.webm" type="video/webm" />
    </video>
</figure>
```

步骤11 完善第二个 section 元素中的内容，具体的代码和 CSS 样式不再显示。

步骤12 通过 footer 元素控制底部内容，在网页底部显示了版权和友情链接等内容。

代码如下：

```
<footer class="clearfix">
   <p>Copyright (c) 2014 abc. <a href="#contact"
       title="Send me a message">Contact me</a>.</p>
   <ul>
      <li><a href="#" title="Validate my CSS">CSS</a></li>
      <li><a href="#" title="Validate my XHTML">XHTML</a></li>
      <li class="printerFriendly">
         <img src="img/printerFriendly.gif">
      </li>
   </ul>
   <p class="thanks">
      Thanks to: <a href="#">Baidu</a>, <a href="#">Google</a>
   </p>
</footer>
```

步骤13 为上个步骤中的元素设置样式，footer 元素的样式代码如下：

```
footer {
    position: relative;
    background-color: #2a1d1b;
    color: #705758;
    padding: 10px 0 10px 0;
    font-size: 75%;
    text-transform: uppercase;
    background-image: url(../img/footerBg.gif);
    background-repeat: repeat-x;
    background-position: bottom;
    border-top: 3px solid #5a353b;
}
```

步骤14 完善网页中的其他内容和样式，网页代码和样式代码不再给出。

步骤15 在浏览器中运行网页，查看效果，最终效果可以参考图 6-31。

6.8 本章习题

1. 填空题

(1) HTML 5 中新增的__________元素用于定义文档的页眉信息。

(2) __________元素用于对网站或应用程序中页面上的内容进行分块。

(3) figcaption 元素用于定义__________元素的标题。

(4) time 元素有两个常用属性，它们分别是__________属性和 pubdate 属性。

(5) details 元素提供了一种替代 JavaScript 的、将画面上局部区域进行展开或收缩的方法，它需要通过__________元素来设置标题。

2. 选择题

(1) __________元素代表文档、页面或应用程序中独立的、完整的、可以独自被外部引用的内容。

A. article　　B. section　　C. aside　　D. figcaption

(2) 下面关于 nav 元素的说法，选项__________是不正确的。

A. nav 元素可以作为传统的导航条

B. nav 元素可用于侧边栏导航

C. menu 元素和 footer 元素都可以用来替换 nav 元素

D. 页内导航和翻页操作时都可以使用 nav 元素

(3) 在显示一篇文章时，这篇文章有两个标题，一个主标题，一个子标题，最好将这两个标题放在__________元素下。

A. aside　　B. hgroup　　C. meter　　D. details

(4) progress 元素常用的两个属性是__________。

A. min 和 max　　B. low 和 high

C. value 和 min　　D. value 和 max

(5) 下面 4 个属性中，__________是 audio 元素的属性。

A. poster　　B. controls　　C. width　　D. height

(6) HTML 5 中新增了一个属性，该属性允许用户编辑元素中的内容。

A. spellcheck　　B. contextmenu　　C. contenteditable　　D. hidden

3. 上机练习

根据效果图构建网页。如图 6-32 所示为一个网页效果，页面包含头部、中间区域和底部 3 个模块，其中头部又包含主标题、副标题和导航链接。读者需要根据 6-32 的效果来设计网页，在合适的位置插入本章介绍的元素。



图 6-32　网页的效果



第 7 章

HTML 5 新型表单的使用

HTML 表单可以用来在网页上显示特定的信息。网站管理者要实现与浏览者之间的沟通，就必须借助于表单这个桥梁。表单通常的应用是用户注册、调查表、搜索界面等。HTML 5 与 HTML 4 相比，在表单方面进行了改进，不仅增加了与表单有关的元素，还增加了与表单和表单域有关的输入类型，本章将介绍 HTML 5 新型表单的使用。在介绍 HTML 5 的新增内容之前，会首先了解一个 HTML 5 中的表单内容。

通过本章的学习，读者不仅可以熟悉表单的基本结构，还可以掌握 HTML 5 中新增的元素、属性和输入类型，也能通过使用 HTML 表单的内容，熟练地构建 HTML 页面。

本章学习目标：

- 了解表单设计时遵循的原则及表单的基本结构
- 掌握 datalist 元素的使用
- 了解 keygen 元素的使用
- 熟悉 output 元素的使用
- 掌握新增的输入类型的使用
- 熟悉新增的两个表单属性
- 掌握新增的与 input 元素有关的属性
- 熟练使用元素、类型和属性构建网页

7.1 了解表单

表单可以用来在网页上显示特定的信息，但主要还是用来收集来自用户的信息，并将收集的信息发送给服务器端处理程序处理。可以说，表单是客户端和服务器端沟通的桥梁，是实现用户与服务器互动的最主要的方式。本节首先了解和回顾表单的基础知识，包括表单的设计原则和基本结构等多个内容。

7.1.1 表单概述

Web 开发者经常会提到网页表单，他们通常所说的“表单”就是指 HTML 表单，一个 HTML 表单是 HTML 文档的一部分，HTML 文档可以包含正常的内容(例如标题、文字和列表等)，也可以包含可视元素(例如文本框、密码框和下拉框等)。

一个好的表单可以为 Web 开发者节省许多工作，如下 3 点列出了设计表单时需要注意的原则：

- 尽量使用下拉列表供用户进行选择，因为列表容易使用，信息也容易处理。
- 如果不能以列表形式提供，那么尽量让用户输入少量文本，这样只需要花费少量的时间，用户易于接受，提供的数据也容易进行处理。
- 只有在必要时才要求用户输入大量文本，因为大量的文本将花费用户很多的时间去填写，也将花费更多的时间去处理。一般情况下，用户是不愿意填写这么多信息的。

目前，表单的交互功能表现在多个方面：输入单行文本、输入多行文本、输入密码，从下拉列表中进行单项选择，从各列项中选择一项或者多项，提交或者取消操作等。例如，在图 7-1 中显示了一个网页的注册页面，该页面的注册信息通过表单来实现。

图 7-1　用户注册页面

7.1.2　表单的基本结构

表单是一个包含表单元素的区域，在网页中负责数据采集功能。一个表单有 3 个基本组成部分——表单元素、表单域和表单按钮。

- 表单元素：这里面包含了处理表单数据所用 CGI 程序的 URL 以及数据提交到服务器的方法。
- 表单域：包含了文本框、密码框、多行文本框、隐藏框、复选框、单选按钮以及下拉选择框和文件上传框等。
- 表单按钮：包括提交按钮、取消按钮和一般按钮，用于将数据传送到服务器上或者取消输入，还可以用表单按钮来控制其他定义了处理脚本的处理工作。

表单使用 form 元素进行定义，它是允许用户在表单中(例如文本框、下拉列表和复选框等)输入信息的元素。在表单中可以添加表单域和表单按钮，基本格式如下：

```
<form action="" enctype="" method="" name="" onsubmit="" onreset="">
    <!-- 添加表单域和表单按钮 -->
</form>
```

<form>标记中可以包含多个元素，上述格式只是列出了几种常用属性，表 7-1 对这些常用属性进行了说明。

表 7-1　<form>标记的常用属性

属性名称	说　明
action	必需属性，用来指定当表单提交时要采取的动作。该属性值一般是要对表单数据进行处理的相关程序地址，也可以是收集表单数据的 E-mail 地址，该 URL 所指向的服务器并不一定要与包含表单的网页是同一服务器，可以是位于任何另外地方的一台服务器，只要给出绝对 URL 地址即可
enctype	设置表单数据的内容类型
method	定义数据传送到服务器的方式，其常用值包括 get(默认值)和 post
name	定义表单的名称
onsubmit	主要是针对 submit 按钮来说的，执行提交操作
onreset	主要是针对 reset 按钮来说的，执行取消操作

可以向 form 元素中添加表单域和表单按钮，这些信息主要是通过 input 元素实现的，而且需要设置该元素的 type 属性，该属性的值决定了不同的表单元素。下面来回顾一下 input 元素 type 属性的取值，这些取值及其说明如表 7-2 所示。

表 7-2　<input>标记的 type 属性值

type 取值	说　明	效 果 图
text	单行输入框	text属性值
password	密码输入框	••••

续表

type 取值	说　明	效 果 图
hidden	隐藏文本框	不显示
file	表示一个文件框，由一个单行文本框和一个“浏览”按钮组成	IE 浏览器：浏览... Chrome 浏览器：选择文件 未选择文件
checkbox	复选框	
radio	单选按钮	
image	表示插入一个图像，作为图形按钮	
submit	提交按钮，将把数据发送到服务器	提交
reset	表示重置按钮，将重置表单数据，以便重新输入	重置
button	普通按钮	普通按钮

7.2　表 单 元 素

HTML 5 新增了 3 种与表单有关的元素，这些元素分别是 datalist、keygen 和 output，下面将分别对这 3 种元素进行介绍。

7.2.1　datalist元素

datalist 元素定义输入框的选项列表，列表通过 datalist 内的 option 元素进行创建。它通常与 input 元素配合使用，来定义 input 可能的取值。在使用<datalist>标记时，需要通过 id 属性为其指定一个唯一的标识，然后为 input 元素指定 list 属性，将该属性值设置为 datalist 元素对应的 id 属性值即可。

【例 7.1】

本例将 input 元素和 datalist 元素结合，完成用户的输入。为了演示 datalist 元素实现的效果，首先向网页中添加以下代码：

```
<form action="#" method="post">
    <table>
      <tr>
          <td align="right">用户名：</td><td><input type="text" ></td>
      </tr>
      <tr>
          <td align="right">密码：</td><td><input type="password"></td>
      </tr>
      <tr>
          <td colspan="2" align="center">
              <input type="submit" value="登录" />
              <input type="reset" /></td>
        </tr>
    </table>
</form>
```

在上述代码中，为表单元素添加了三行两列的表格元素，其中前两行向用户提供用户名和密码，最后一行提供操作信息。在浏览器中运行上述代码进行测试，点击用户名输入框，此时没有任何效果，如图 7-2 所示。



图 7-2 用户登录页面

重新更改上述代码，向表单中添加 datalist 元素列表，并且将其绑定到用户名所在的 input 元素。代码如下：

```
<input type="text" list="namelist">
<datalist id="namelist">
   <option>admin</option>
   <option>lucy</option>
   <option>FoverMe</option>
</datalist>
```

重新运行上述代码，观察 datalist 元素的效果，如图 7-3 所示。在该图所示的界面中，用户可以手动输入用户名，也可以选择提供的用户名。



图 7-3 datalist元素的效果

7.2.2 keygen元素

keygen 元素是密钥生成器，作用是提供一种验证用户的可靠方法。当提交表单时会生成两个键：一个是私钥，它存储在客户端；一个是公钥，它被发送到服务器。其中，公钥可用于验证用户的客户端证书。

<keygen>标记可以使用多个属性，常用属性及其说明如表 7-3 所示。

表 7-3　<keygen>标记的常用属性

属性名称	说　明
autofocus	使 keygen 字段在页面加载时获得焦点
challenge	如果使用，则将 keygen 的值设置为在提交时间
disabled	禁用 keytag 字段
form	定义该 keygen 字段所属的一个或者多个表单
keytype	定义 keytype。rsa 生成 RSA 密钥
name	定义 keygen 元素的唯一名称。name 属性用于在提交表单时搜集字段的值

【例 7.2】

继续在上个示例的基础上添加代码，演示 keygen 元素的使用。相关代码如下：

```
<keygen name="security"></keygen>
```

在浏览器中运行上述代码，查看效果，如图 7-4 所示。

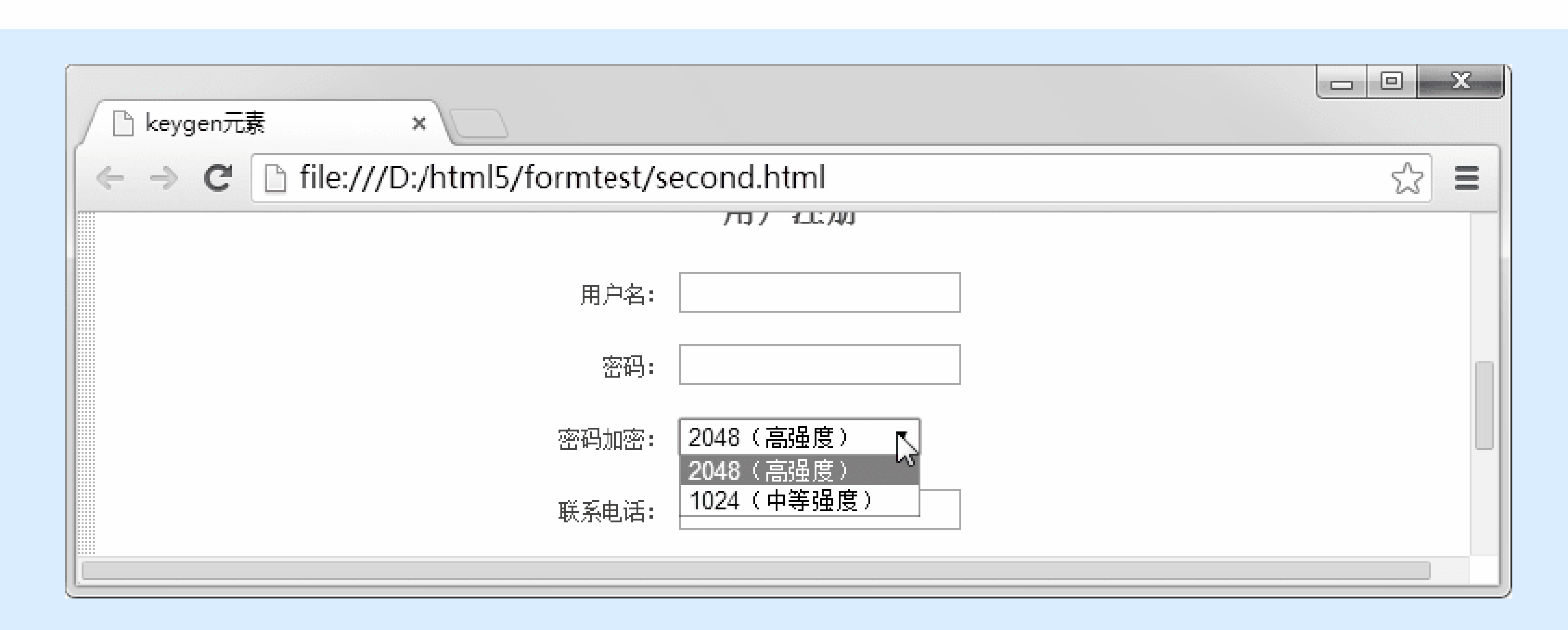


图 7-4　keygen元素的使用

7.2.3　output元素

output 元素用于不同类型的输出，例如脚本输出。<output>标记的常用属性有 3 个，说明如下。

- for：定义输出域相关的一个或者多个元素。
- form：定义输入字段所属的一个或者多个表单。
- name：定义对象的唯一名称(表单提交时使用)。

【例 7.3】

通常情况下，会将 output 元素与 JavaScript 脚本结合使用。一般步骤如下。

步骤 01　向 HTML 网页中添加表单，在表单中添加两行两列的表格，第一行用于输出 23+12 的结果，第二行用于执行操作。部分代码如下：

```
<form action="first.html" method="post">
    <table width="300" height="100" align="center">
    <caption><h1>output 元素</h1></caption>
        <tr>
```

```
            <td  align="right">
              结果：</td><td>23+12=<output name="result"/></td>
        </tr>
        <tr>
            <td colspan="2" align="center">
                <input type="button" onClick="javascript:GetResult()"
                  value="计算" /></td>
        </tr>
    </table>
</form>
```

步骤 02 从上述代码中可以看出，通过 output 元素输出结果，并且为操作按钮添加了 onClick 事件属性，当单击按钮时，会调用 GetResult()脚本函数。该函数的代码如下：

```
<script>
     function GetResult(){
         var form = document.forms[0];
         form['result'].value = 25;
         event.preventDefault();
     }
</script>
```

步骤 03 在浏览器中运行上述代码，查看效果。

7.3 输 入 类 型

HTML 5 中新增的输入类型为用户提供了更好的输入控制和数据验证，本节将介绍这些新增的输入类型。

7.3.1 email类型

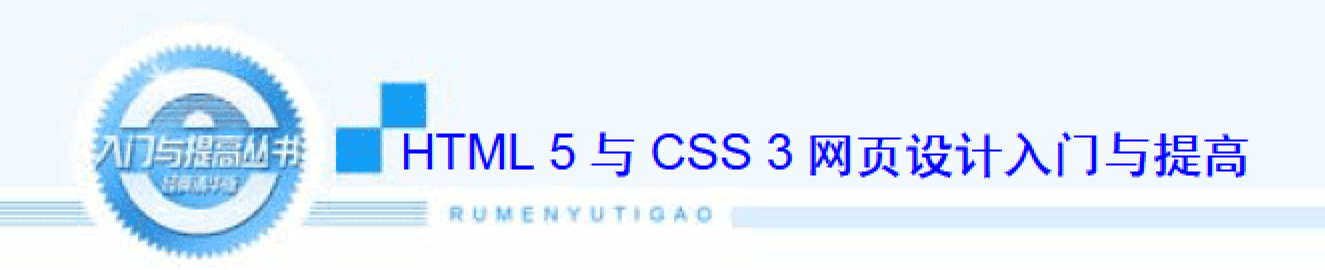
email 类型用于应该包含 E-mail 地址的输入文本框，该文本框与其他文本框在页面显示时没有区别，专门用于接收 E-mail 地址信息。在提交表单时，会自动验证文本框中的内容是否符合 E-mail 邮件地址格式；如果不符合，将提示相应的错误信息。

注意：无论是本节介绍的 email 类型，还是后面几节介绍的其他类型，它们都不会自动验证输入框是否为空，而是在不为空的情况下验证用户输入的内容是否符合标准。简单地说，只有在输入框的内容不为空时，这些类型的输入框才会执行验证功能。

【例 7.4】

目前，许多网站要求在用户注册时输入有效的电子邮箱，这时可以直接将 input 元素的 type 属性值指定为 email。例如，直接向表单元素中添加一个 email 类型的 input 元素和提交按钮，页面代码如下：

```
<form action="#" method="get">
    发件人：<input type="email" name="frommail" /><br/>
```




```
    <input type="submit" value="提交" />
</form>
```

在浏览器中运行上述代码并向页面中输入内容进行测试，不同的浏览器可能导致效果有所不同。例如，图 7-5 和 7-6 分别为 Chrome 浏览器和 Firefox 浏览器中的效果。



图 7-5　Chrome浏览器中的效果

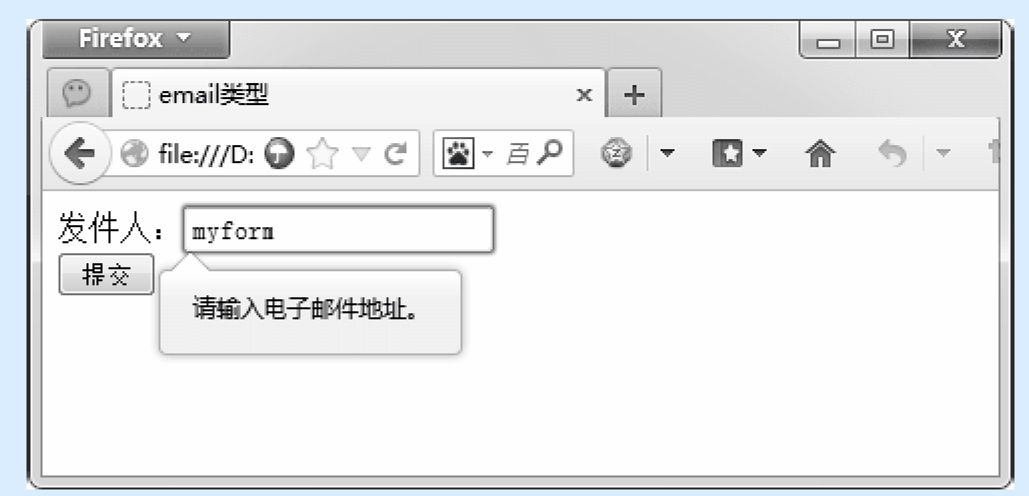


图 7-6　Firefox浏览器中的效果

提示：在某些情况下，并不只是要求用户输入一个电子邮箱，如果用户存在多个邮箱，那么可以允许用户输入多个。将邮箱地址输入框的 multiple 属性的值设置为 true 时，允许用户输入一串逗号分隔的邮箱地址。

7.3.2　url类型

url 类型用于应该包含绝对 URL 地址的输入框。在提交表单时，会自动验证用户输入 url 文本框中的值，如果输入的值不合规则，不允许提交，并且会有提示信息。url 类型的输入框适用于多种情况，例如个人主页、百度地址和博客地址等。

【例 7.5】

url 类型与 email 类型的使用方式一样，如下为 url 的基本使用代码：

```
<form action="#" method="get">
    博客主页：<input type="url" /><br/>
    <input type="submit" value="提交" />
</form>
```

在浏览器中运行上述代码并输入内容进行测试，图 7-7 和 7-8 分别为 Chrome 浏览器和 Firefox 浏览器中的效果。



图 7-7　Chrome浏览器



图 7-8　Firefox浏览器

不同的浏览器对 url 类型的输入框的要求有所不同，例如，在图 7-7 和 7-8 所示的浏览器中，要求用户必须输入完整的 URL 地址，并且允许地址前有空格的存在。

7.3.3 number类型

number 类型用于包含数字的输入框。在提交表单时，会自动检查该输入框的内容是否为数字。当使用的浏览器不支持 number 类型时，会自动显示为一个普通的输入框。

【例 7.6】

例如，直接向页面中添加 number 类型的 input 元素。代码如下：

```
<form action="#" method="get">
    您的年龄：<input type="number" value="20" /><br/>
    <input type="submit" value="提交" />
</form>
```

在浏览器中运行上述代码，如图 7-9 和 7-10 所示分别为 Chrome 浏览器和 Opera 浏览器中的显示效果。

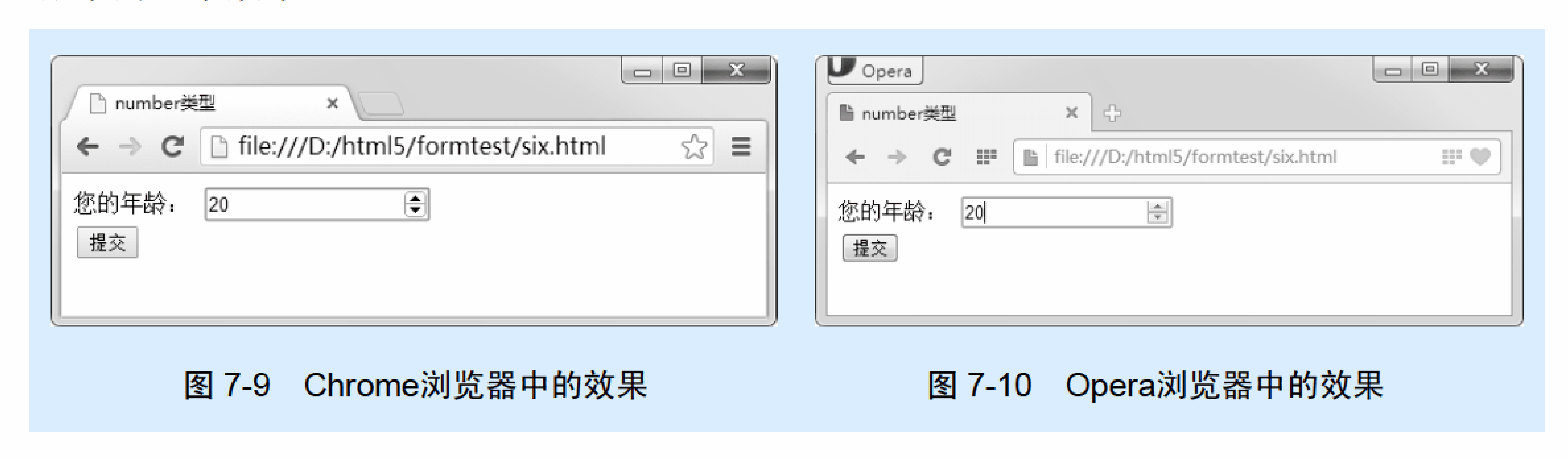


图 7-9 Chrome浏览器中的效果　　图 7-10 Opera浏览器中的效果

如果可以向输入框中手动输入数值，也可以点击输入框的数值按钮进行控制。如果用户输入的值不合法，则会进行验证，如图 7-11 和 7-12 所示为验证时的效果。

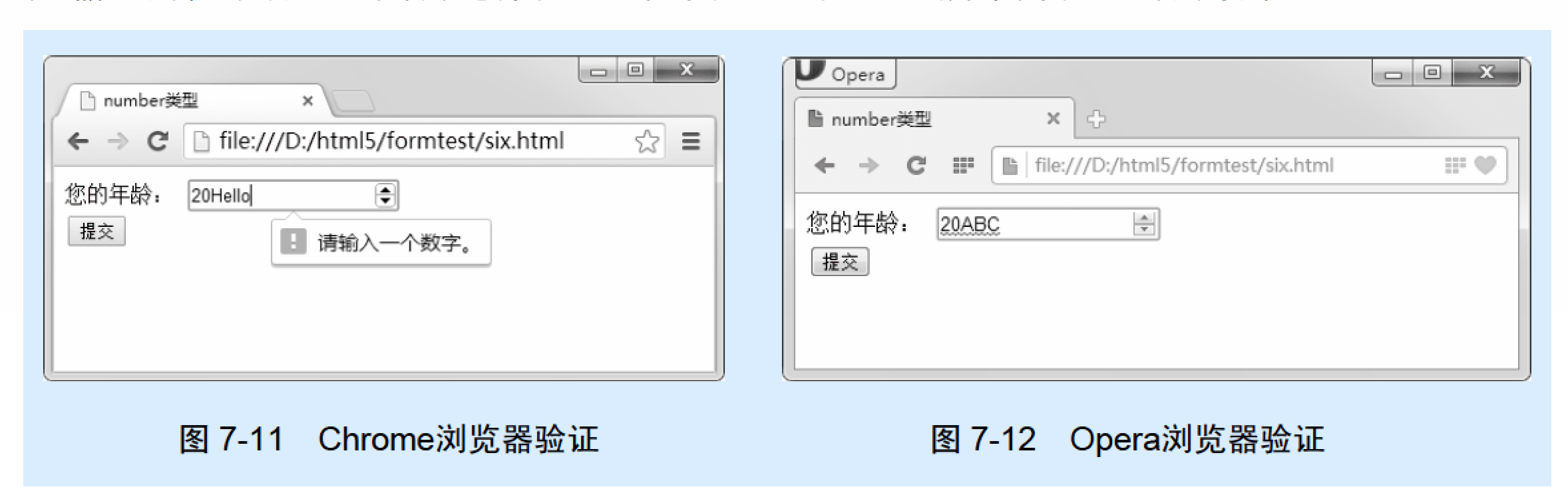


图 7-11 Chrome浏览器验证　　图 7-12 Opera浏览器验证

number 类型的输入框能够设置对所接受的数字的限定，除了 value 属性外，还可以和其他的属性结合使用，这些属性的说明如下所示。

- max：指定输入框可以接受的最大的输入值。
- min：指定输入框可以接受的最小的输入值。
- step：输入域合法的间隔，如果不设置，默认值是 1。

【例 7.7】

重新对上个示例中的内容进行更改，分别设置 min、max 和 step 属性的值。页面代码如下：

```
<form action="#" method="get">
  您的年龄：<input type="number" value="20" min="10" max="110" step="5" />
  <br/>
  <input type="submit" value="提交" />
</form>
```

在浏览器中运行上述代码，选择数值按钮或者输入内容进行测试，运行效果如图 7-13 和 7-14 所示。



图 7-13　Chrome浏览器测试效果　　图 7-14　Opera浏览器单击数值按钮

7.3.4 range类型

range 类型用于应该包含一定范围内数值的输入范围，并且也可以设定对所接受值的限定。它的常用属性与 number 类型一样，通过 min 属性和 max 属性，可以设置最小值与最大值(默认值分别为 0 和 100)，通过 step 属性指定每次拖动的步幅。

【例 7.8】

如下代码显示了 range 类型的 input 元素的使用，并且为其指定 min、max 和 step 属性的值。相关步骤如下。

步骤 01 创建 HTML 网页，并向该页面中分别添加 range 类型和 text 类型的 input 元素，前者指定输入范围，后者显示前一个输入框的值。相关内容如下：

```
<form action="#" method="get">
    价格范围:
    <input type="range" min="10" max="100" step="5" name="price"
      onChange="GetValue()"/><br/>
    <input type="text" name="checkvalue" disabled readonly />
</form>
```

步骤 02 添加 GetValue()函数的脚本代码，在这段代码中设置 name 属性值是 checkvalue 的输入框。代码如下：

```
<script>
function GetValue(){
    var form = document.forms[0];
    form['checkvalue'].value= form['price'].value;
}
```

```
</script>
```

步骤 03 如果浏览器不支持 range 类型，那么会在页面中显示一个普通输入框。在支持 range 类型的浏览器中，range 类型的输入框通常以滑动条的方式进行值的指定。例如，图 7-15 和 7-16 分别为 Chrome 浏览器和 Firefox 浏览器中的效果。



图 7-15 Chrome浏览器中的效果　　图 7-16 Firefox浏览器中的效果

7.3.5 datepickers类型

datepickers 类型是指日期类型，HTML 中提供了多个可供选取日期和时间的新输入类型，用于验证输入的日期。

表 7-4 对 HTML 5 中新增的日期和时间输入类型做了具体的说明。

表 7-4 HTML 5 中新增的日期和时间类型

日期和时间类型	说 明
date	选取日、月、年
month	选取月、年
week	选取周和年
time	选取时间(小时和分钟)
datetime	选取时间、日、月、年(UTC 时间)
datetime-local	选取时间、日、月、年(本地时间)

如果浏览器不支持表 7-4 中列举的日期和时间类型，那么该类型的输入框在网页中显示为一个普通输入框。

【例 7.9】

向 HTML 网页中添加多个 input 元素，分别指定这些元素的 type 属性值，将其指定为表 7-4 中的类型。页面代码如下：

```
<form action="#" method="get">
    <table height="150" width="450">
        <tr>
            <td><input type="date" /></td>
            <td><input type="month" /></td>
        </tr>
        <tr>
            <td><input type="week" /></td>
            <td><input type="time" /></td>
        </tr>
```

```
        <tr>
            <td><input type="datetime" /></td>
            <td><input type="datetime-local" /></td>
        </tr>
    </table>
</form>
```

在浏览器中运行上述代码，观察效果，图 7-17 和 7-18 分别展示了 Chrome 浏览器和 Opera 浏览器的效果。

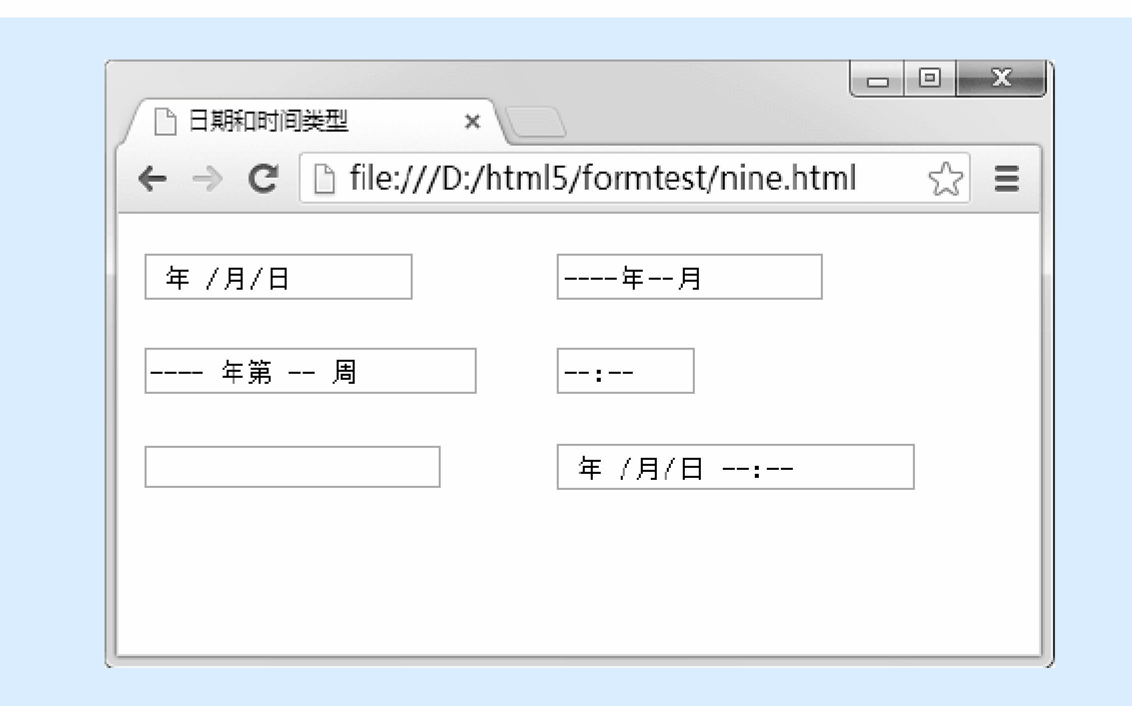


图 7-17　Chrome浏览器的初始效果

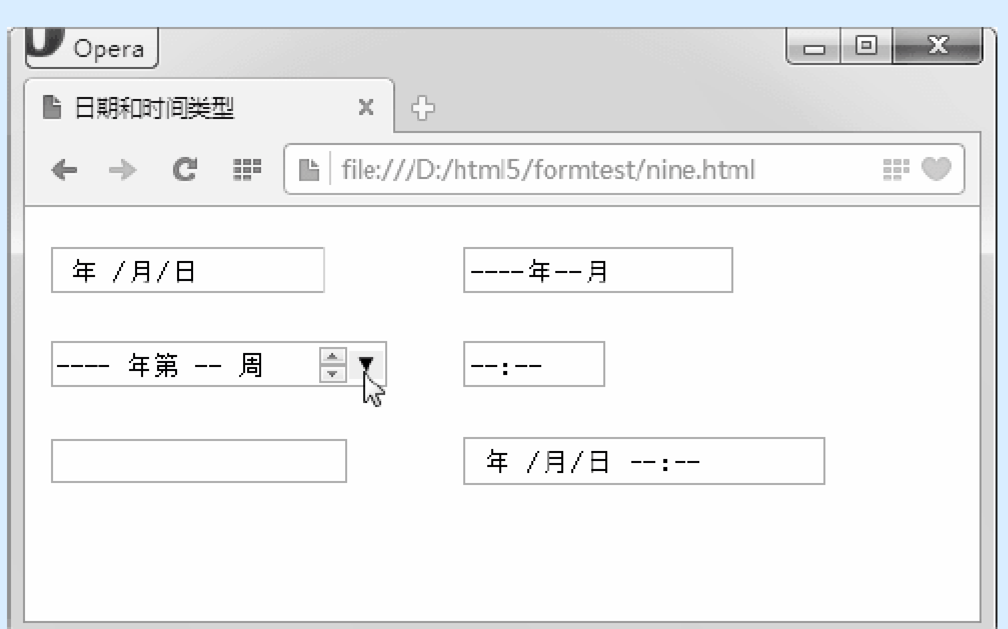


图 7-18　Opera浏览器鼠标悬浮效果

用户可以向输入框中输入内容，也可以单击输入框之后的按钮进行选择，图 7-19 为 Chrome 浏览器选择年、月、日时的效果，图 7-20 为 Opera 浏览器选择年和周的效果。

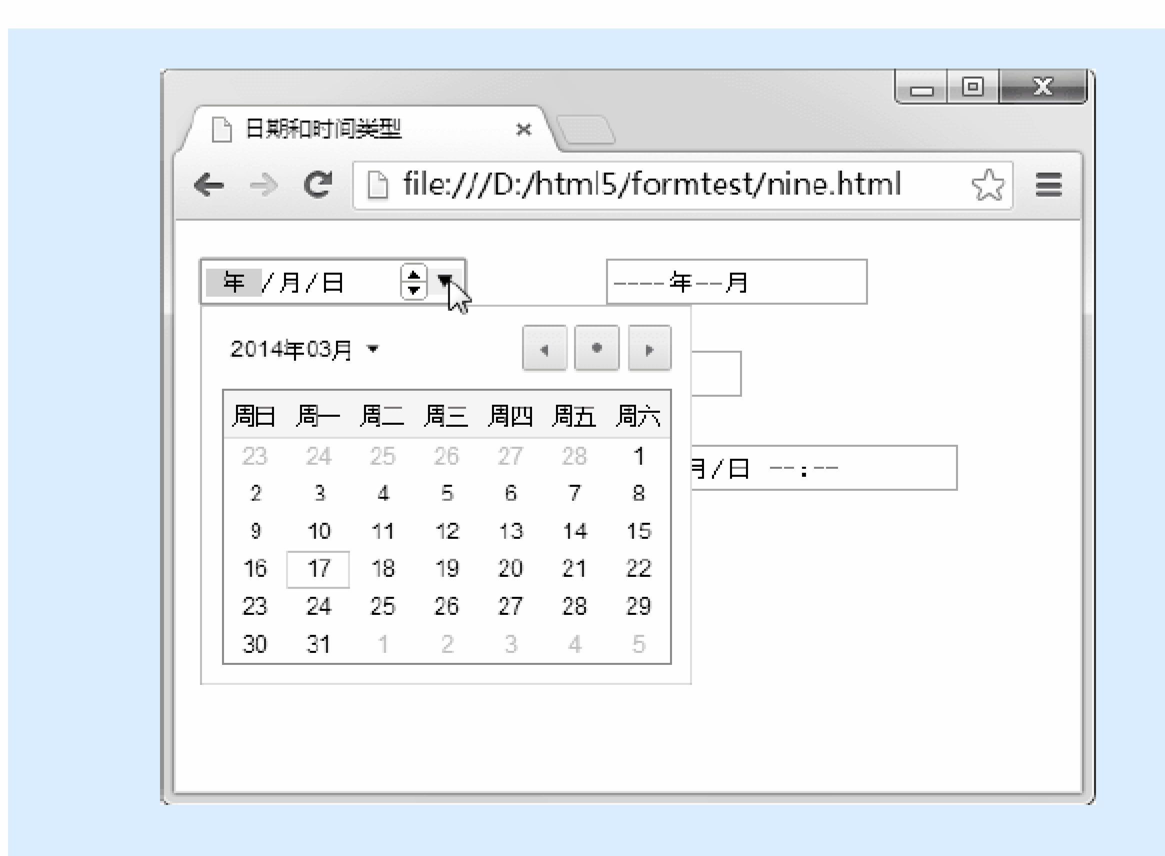


图 7-19　选择年、月、日

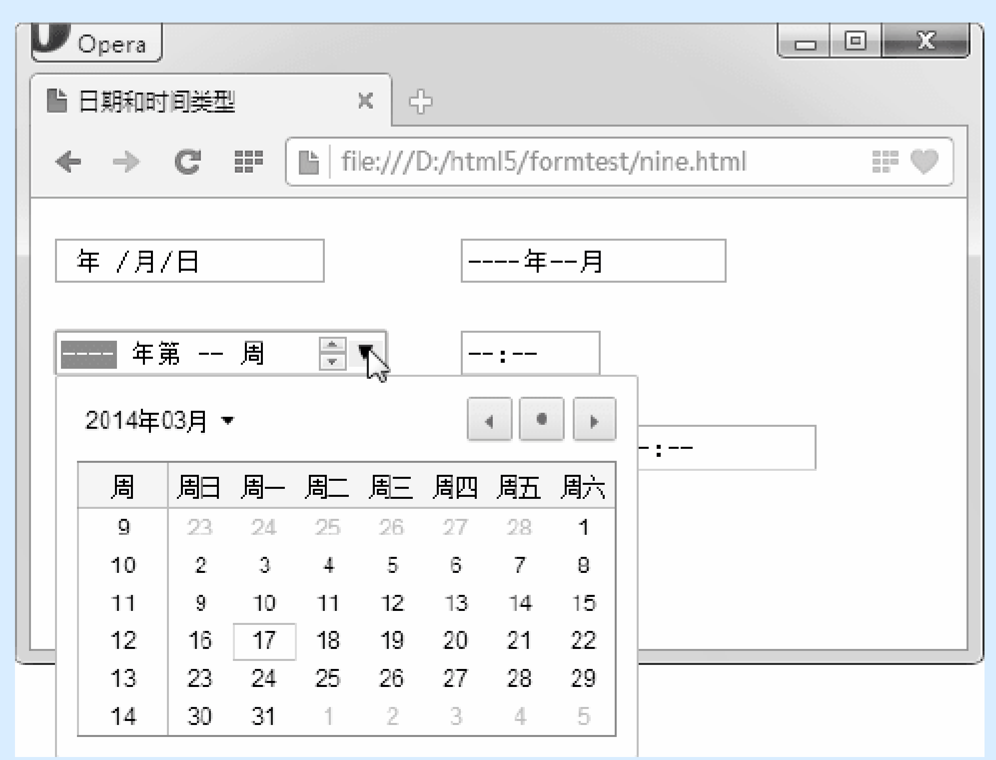


图 7-20　选择年和周

7.3.6 search类型

search 类型是一种专门用于输入搜索关键词的输入框，它能自动记住一些字符，例如站点搜索或者 Google 搜索。如果浏览器不支持 search 输入类型，那么会在网页中显示为一个普通输入框；如果浏览器支持 search 输入类型，在用户输入内容后，会在右侧附带一个删除图标的按钮，单击这个图标按钮可以快速清除内容。

【例 7.10】

向 HTML 表单元素中添加一个 search 类型的输入框和一个提交按钮。页面代码如下：

```
<form action="#" method="get">
   <input type="search" name="searchinfo" />
   <input type="submit" vaule="提交" />
</form>
```

在浏览器中运行本示例，图 7-21 为 Chrome 浏览器运行时的初始效果，图 7-22 为输入内容时的效果。

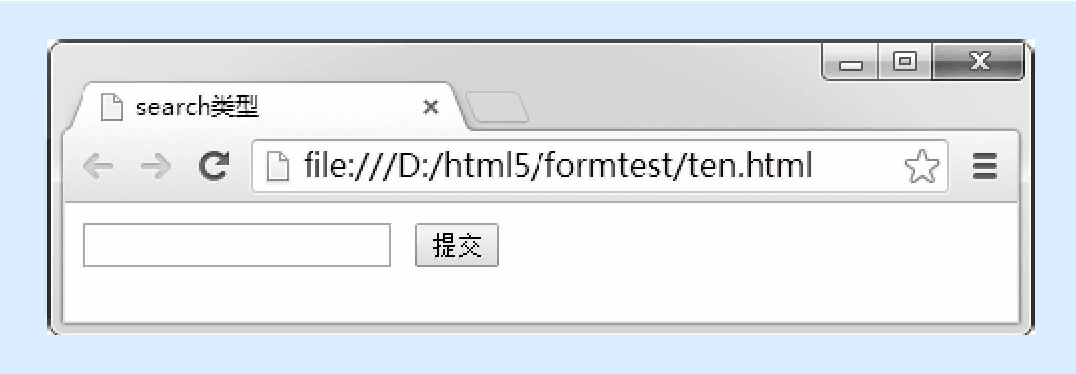


图 7-21　初始效果

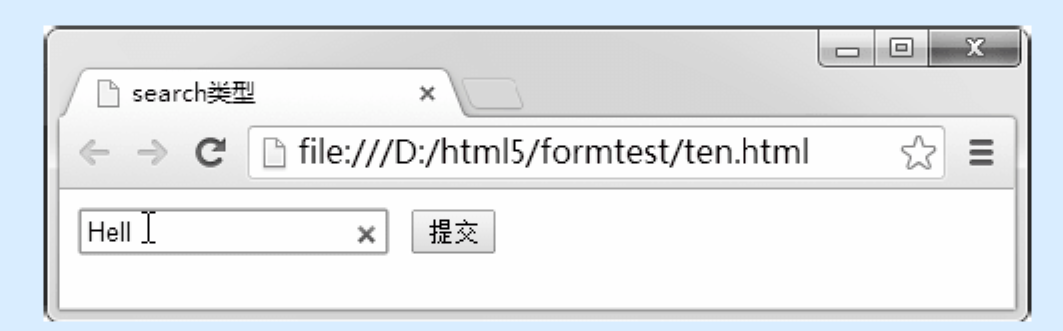


图 7-22　输入内容时的效果

7.3.7 color类型

color 类型用于实现一个 RGB 颜色输入，其基本形式是#RRGGBB，默认值是#000000。如果浏览器支持 color 类型，那么会向用户提供一个颜色选取器，用户在单击时能够弹出一个颜色选盘供其选择。如果浏览器不支持 color 类型，那么会自动地在网页中显示为一个普通输入框。

【例 7.11】

向 HTML 网页中添加两个 color 输入类型的 input 元素，第一个输入元素使用默认的 value 值，第二个重新指定 value 属性值，然后添加一个提交按钮。页面代码如下：

```
<form action="#" method="get">
   <input name="color1" type="color" />
   <input name="color2" type="color" value="#FF3E96" />
   <input type="submit" value="提交" />
</form>
```

在浏览器中运行上述代码，图 7-23 为 Chrome 浏览器支持 color 输入类型的效果，而图 7-24 为 Firefox 浏览器不支持 color 类型时的效果。

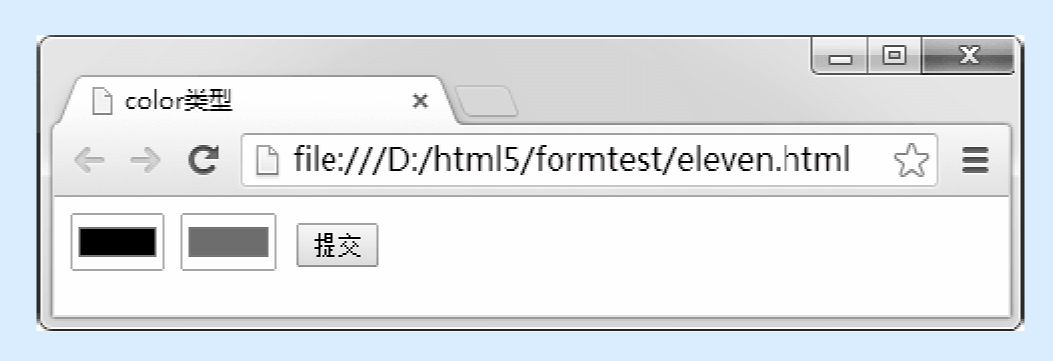


图 7-23　Chrome浏览器中的效果

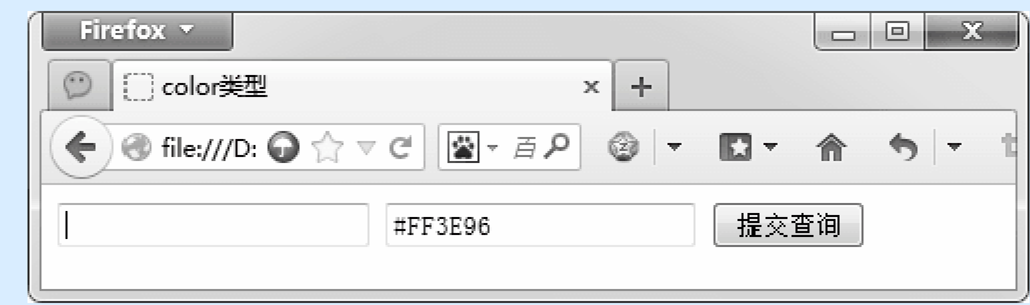


图 7-24　Firefox浏览器不支持

在支持 color 输入类型的 Chrome 浏览器中，直接单击颜色，弹出如图 7-25 所示的颜色选取器。在颜色选取器中，用户可以选择图中左侧的基本颜色，也可以在右侧提供的颜色中进行选择，选择后单击“添加到自定义颜色”按钮，这时会在自定义颜色中显示自定义的颜色，如图 7-26 所示。



图 7-25 “颜色”对话框

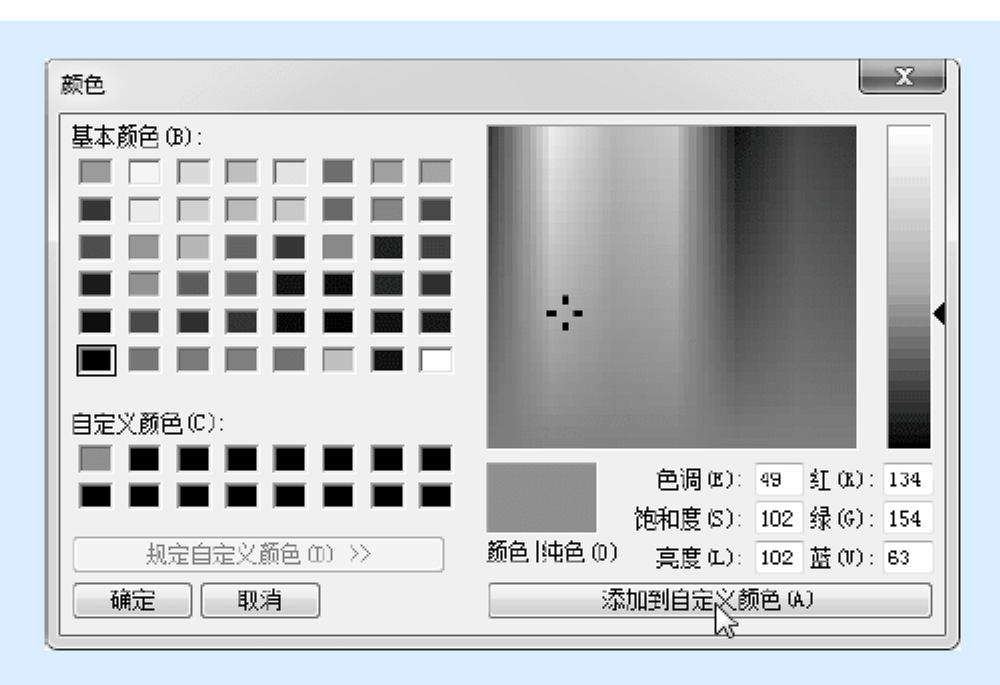


图 7-26 添加自定义颜色

7.3.8 tel类型

tel 类型用于从语义上实现一个电话号码的输入，但是电话号码的格式千差万别，很难实现一个通用的格式。因此，tel 类型通常会和 pattern 属性配合使用，pattern 属性的详细内容会在 7.4.2 节进行介绍。

【例 7.12】

向 HTML 网页添加表示手机号码的 tel 输入类型，并通过 pattern 属性指定电话号码的格式。代码如下：

```
<form action="#" method="get">
    手机号码：<input name="telphone" type="tel" pattern="^\d{11}$" />
    <input type="submit" value="提交" />
</form>
```

在浏览器中运行上述代码，向页面输入内容进行测试，图 7-27 和 7-28 分别为 Chrome 浏览器和 Firefox 浏览器的测试效果。

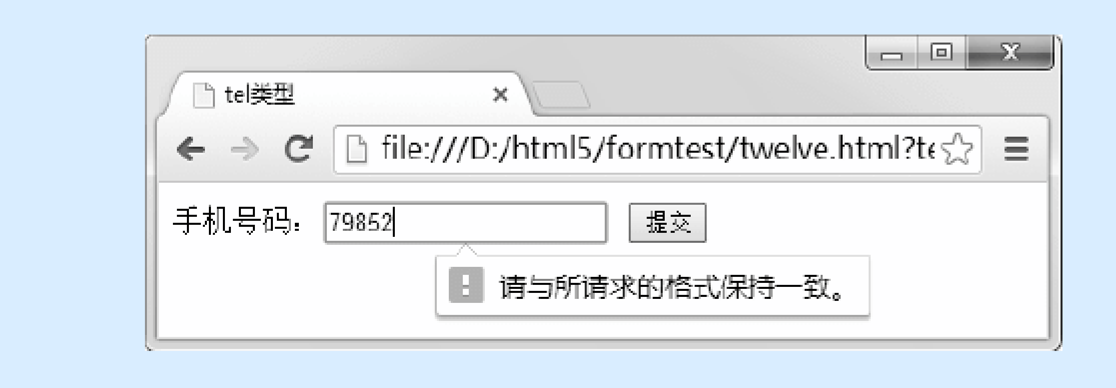


图 7-27 Chrome浏览器中的效果

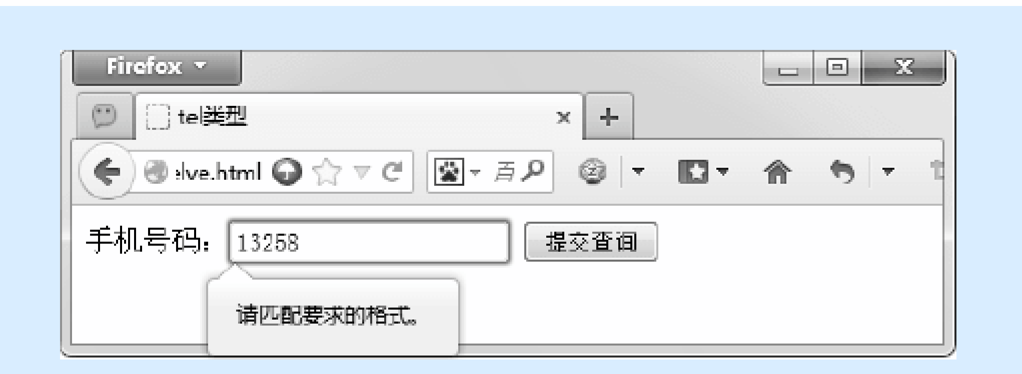


图 7-28 Firefox浏览器中的效果

7.4 表 单 属 性

HTML 5 中新增了一系列与表单有关的属性，本节将介绍 HTML 5 中的新增属性，首先介绍表单的新增属性，接着再介绍与表单的 input 元素有关的新增属性。

7.4.1 表单属性

简单地说，表单属性就是与 form 元素有关的属性。HTML 5 中新增的、与表单有关的

两个表单属性是 autocomplete 属性和 novalidate 属性。

1. autocomplete属性

atuocomplete 属性指定所有的表单控件是否应该拥有自动完成功能。该属性的值包含 on 和 off：如果将属性值指定为 on，那么表示执行自动完成功能；如果将属性值指定为 off，那么表示关闭自动完成功能。

【例 7.13】

为页面中的<form>标记指定 autocomplete 属性，并且将该属性的值指定为 on，然后向表单中添加两个输入框和一个按钮。页面代码如下：

```
<form id="formOne" autocomplete="on">
    用户名：<input  type="text" id="autoFirst" name="autoFirst"/><br/><br/>
    昵   称: <input type="text" id="autoSecond"
      name="autoSecond" /><br/><br/>
    <input type="submit" value="提交" />
</form>
```

向页面中输入内容，进行提交，测试时的效果如图 7-29 和 7-30 所示。

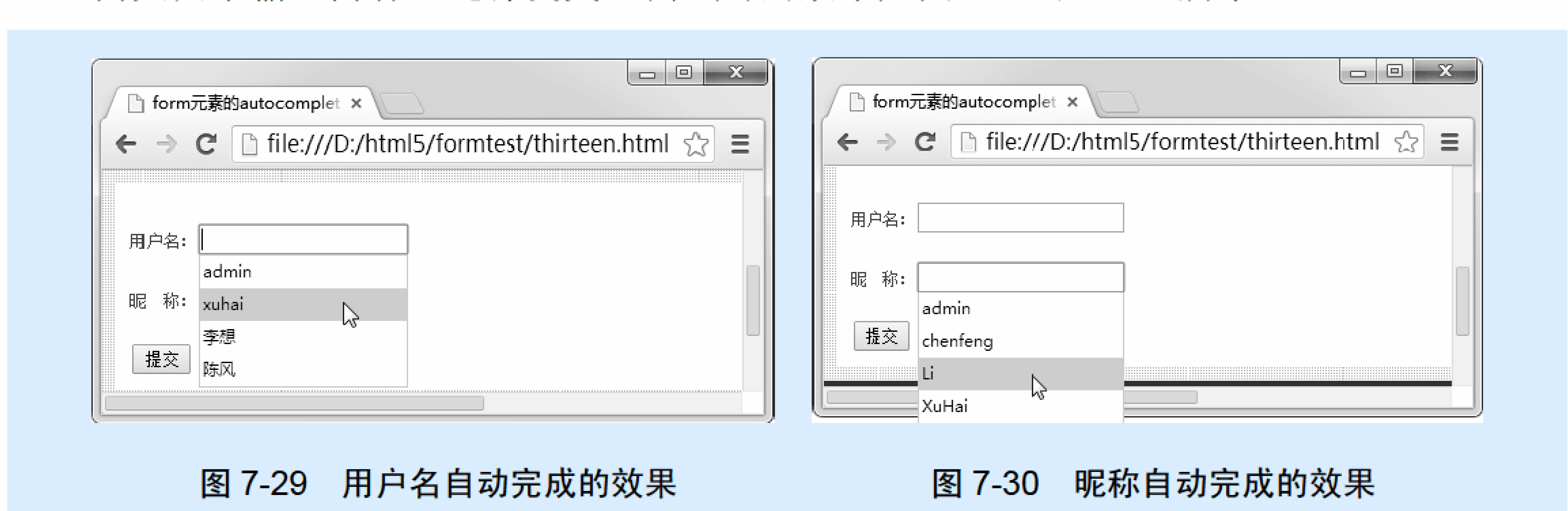


图 7-29 用户名自动完成的效果　　图 7-30 昵称自动完成的效果

浏览器的不同可能会导致显示 autocomplete 属性的效果会有所不同，图 7-31 和 7-32 分别显示了 Firefox 浏览器和 Opera 浏览器的效果。

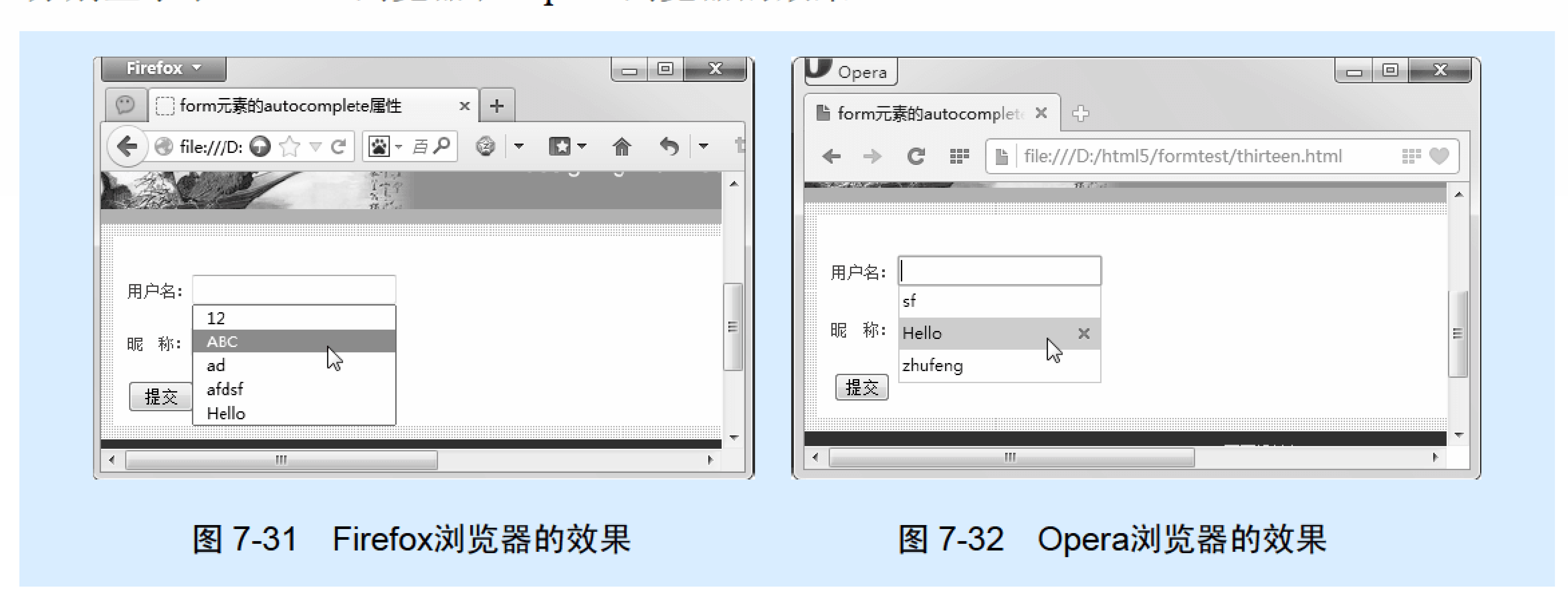


图 7-31 Firefox浏览器的效果　　图 7-32 Opera浏览器的效果

autocomplete 属性不仅可以用于 form 元素，还可以用于所有类型的 input 元素。可以重新更改例 7.13 中的代码，将提供用户输入“昵称”的文本框的 autocomplete 属性的值设

置为 off。这时页面代码如下：

```
<form id="formOne" autocomplete="on">
   用户名：<input  type="text" id="autoFirst" name="autoFirst"/><br/><br/>
   昵   称：<input type="text" autocomplete="off"
     id="autoSecond" name= "autoSecond" /><br/><br/>
   <input type="submit" value="提交" />
</form>
```

在上述代码中，为<form>标记指定 autocomplete 属性的值为 on，这表示 form 元素中的两个输入框都实现了自动完成功能。同时，又为“昵称”输入框指定了 autocomplete 属性，并且该属性的值为 off，这表示该输入框不开启自动完成功能。

重新在浏览器中运行上述代码进行测试，观察 autocomplete 属性的实现效果，图 7-33 和 7-34 分别为两个输入框的效果。从这两个图中可以发现，“用户名”输入框依旧会实现自动完成功能，而“昵称”输入框关闭了自动完成功能。

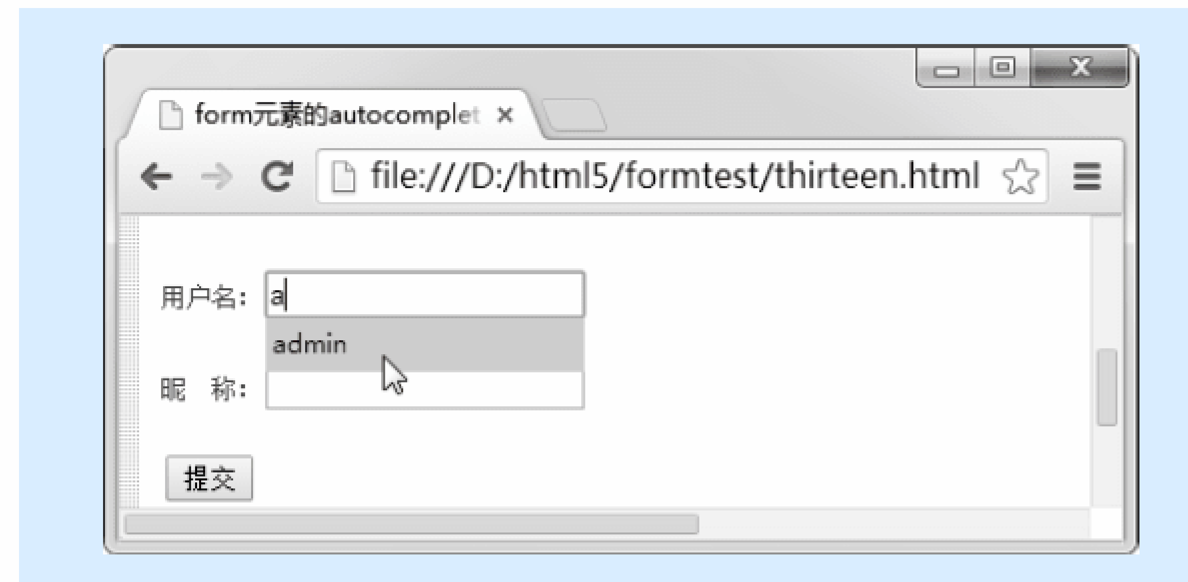



图 7-33　“用户名”输入框自动完成

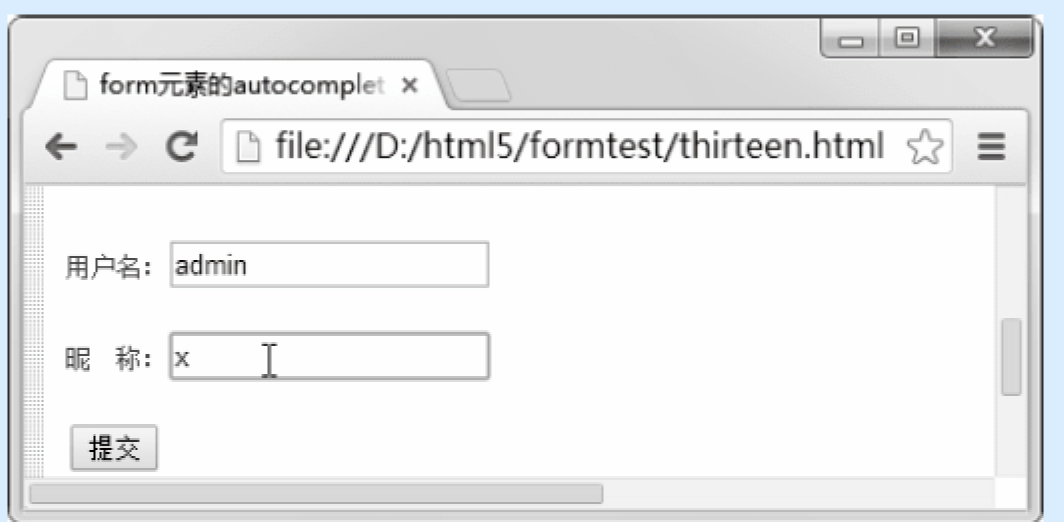


图 7-34　“昵称”输入框关闭了自动完成

技巧：各个浏览器对 autocomplete 属性支持的列表个数有差异，例如 Google 浏览器和 Opera 浏览器记忆前 6 个，按输入顺序先后显示，超出之后按照输入值提交的次数降序排列 6 个。Firefox 浏览器自动完成列表数据的个数没有限制，按提交的先后顺序，超出则滚动显示。

无论是 form 元素还是多个类型的 input 元素，都可以使用 autocomplete 属性，使用该属性时的注意事项如下：

- input 元素应该位于 form 表单中，并且应该含有 submit 提交按钮，要指定 name 属性。
- 默认情况下，text 类型的 input 元素含有 autocomplete 为 on 的属性，如果要取消自动完成，那么需要将 autocomplete 指定为 off。
- 浏览器自动记忆的值为已提交的值，并且这些值的顺序为提交的先后顺序。
- 浏览器自动记忆的值，是以标记的 name 属性为标准的，也就是说，如果表单中的 input 标记有相同的 name，那么就有相同的自动完成列表(与是否在同一个 form 中和是否在同一个网页中无关)。
- 自动完成列表信息没有存放在浏览器缓存中。

2. novalidate属性

novalidate 属性指定在提交表单时不对表单进行有效的检查。为表单设置该属性时，可以关闭整个表单验证，这样可以使 form 内的所有表单控件不进行验证。简单地说，novalidate 属性用于取消表单验证。

除了指定 form 元素的 novalidate 属性取消验证外，还有一种取消验证的方法，即利用 input 元素的 formnovalidate 属性，这样可以控制单个表单控件的验证。利用 input 元素的 formnovalidate 属性可以让表单验证对单个 input 元素失效，例如，表单的第二部分需要验证的内容数量很少时，可以通过这种方法为 submit 类型的按钮指定 formnovalidate 属性，在用户单击按钮时，相当于利用 form 元素的 novalidate 属性，这会导致整个表单都失效。

7.4.2 input属性

HTML 5 中提供了一系列与 input 元素有关的属性，为 input 元素指定这些属性可以实现不同的功能。具体来说，HTML 5 新增的与 input 元素有关的属性包括 autocomplete 属性、autofocus 属性、form 属性、表单重写属性(formaction、formenctype、formmethod、formnovalidate 和 formtarget)、width 和 height 属性、list 属性、min、max 和 step 属性、multiple 属性、pattern 属性、placeholder 属性以及 required 属性。

在上述属性中，有些属性在前面已经提到并使用过，例如 autocomplete 属性、list 属性、以及 min、max 和 step 属性。下面介绍几种常用的新增属性。

1. autofocus属性

autofocus 属性指定页面加载后是否自动获取焦点，一个页面上只能有一个元素指定 autofocus 属性。autofocus 属性适用于所有类型的<input>标记，该属性的值是一个布尔值，将标记的属性值指定为 true 时，表示页面加载完毕后会自动获取该焦点。

【例 7.14】

在例 7.13 的基础上进行更改，为提供用户输入的“昵称”文本框指定 autofocus 属性。代码如下：

```
昵   称:
<input type="text" autocomplete="off" id="autoSecond" name="autoSecond"
autofocus="true" />
```

在浏览器中运行页面，查看效果，初始效果如图 7-35 所示。从该图中可以看出，页面加载完毕后直接将焦点放到“昵称”输入框中。

2. multiple属性

multiple 属性指定输入框可以选择多个值，该属性适用于 email 和 file 类型的<input>标记。multiple 属性用于 email 类型的<input>标记时，表示可以向文本框中输入多个 E-mail 地址，多个地址之间通过逗号进行分隔；multiple 属性用于 file 类型的<input>标记时，表示可以选择多个文件。

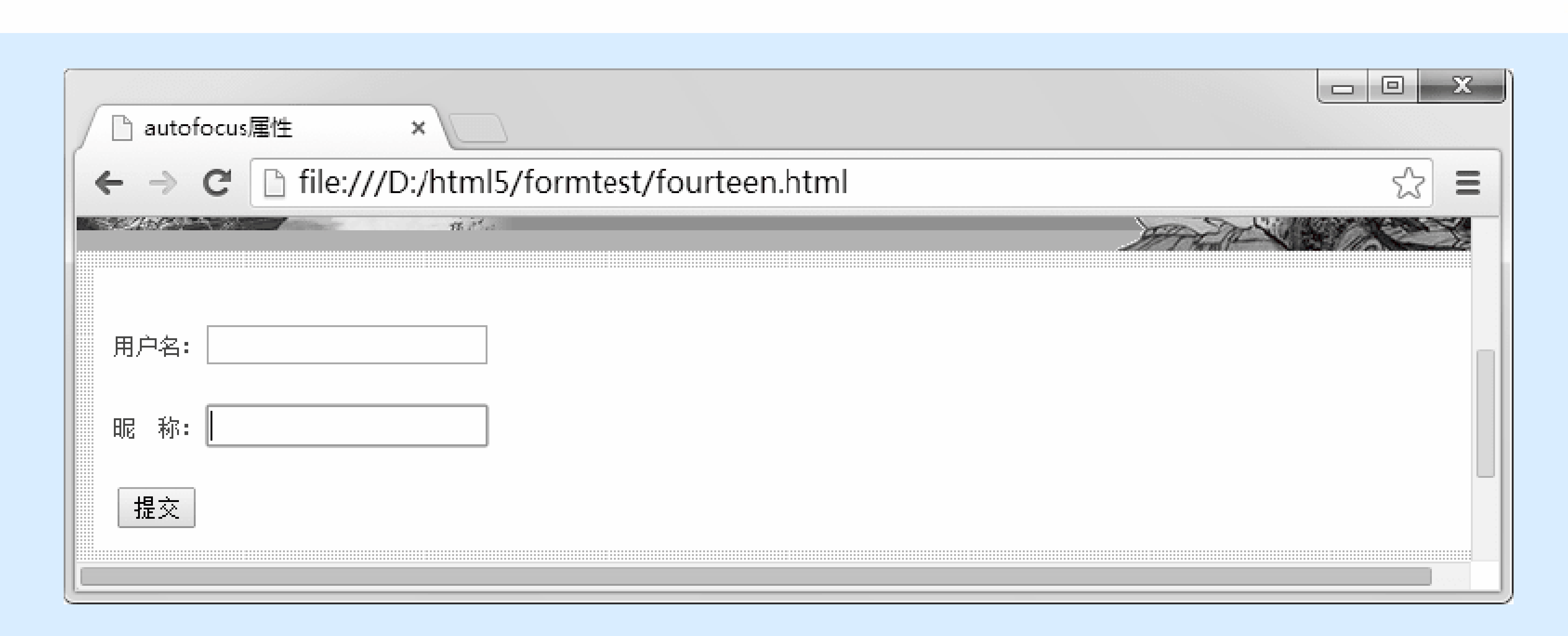


图 7-35　autofocus属性的使用

【例 7.15】

本示例演示 email 类型和 file 类型的<input>标记的 multiple 属性的使用。基本实现步骤如下。

步骤 01　向 HTML 页面中添加两个<input>标记，分别指定该标记的 type 属性为 email 和 file。代码如下：

```
<form id="formOne">
    电子邮箱：<input  type="email" name="myemail"/>  (如果电子邮箱有多个，请使用逗号进行分隔)<br/><br/>
    上传照片：<input type="file" name="selfile" /><br/><br/>
    <input type="submit" value="提交" />
</form>
```

步骤 02　在浏览器中运行上述页面，查看效果，如图 7-36 所示为输入多个电子邮箱地址时的效果。

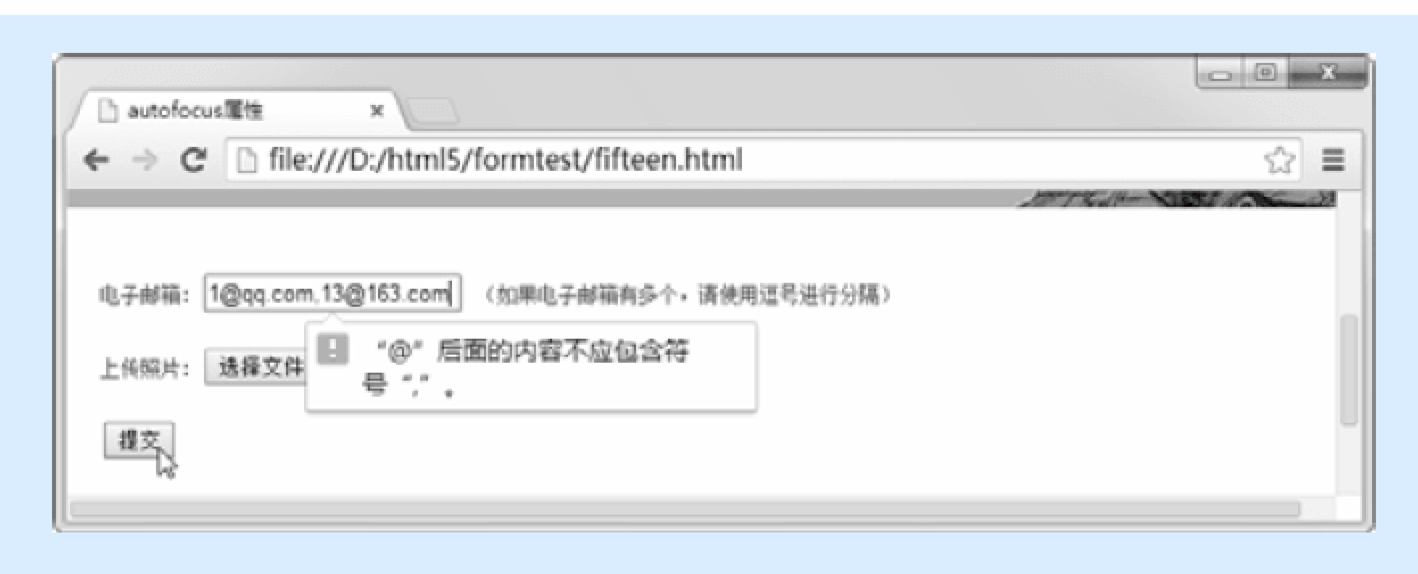


图 7-36　输入多个电子邮箱地址时的效果

步骤 03　从图 7-36 中可以看出，不指定 multiple 属性而输入多个电子邮箱时，会自动验证并提示错误。这时，重新更改步骤 01 中的代码，为两个<input>标记添加 multiple 属性。代码如下：

```
电子邮箱：<input  type="email" name="myemail"
multiple="true"/>  (如果电子邮箱有多个，请使用逗号进行分隔)
上传照片：<input type="file" name="selfile" multiple />
```

步骤 04　重新运行上述代码，查看效果，并输入内容进行测试。如果要实现多张照片的选择，可以按下 Shift 键再选择图片。

3. pattern属性

在前面介绍 tel 输入类型时已经使用到了 pattern 属性，该属性用于验证 input 框的模式，模式即正则表达式。pattern 属性适用于类型是 text、search、url、tel、email 和 password 的<input>标记。

【例 7.16】

向网页的表单中添加多行两列的表格，每一行供用户输入不同的信息，通过 pattern 属性指定验证。实现步骤如下。

步骤 01 第一行第二列供用户输入姓名，要求用户的姓名必须是汉字，而且长度大于 12，小于 2。代码如下：

```
<td align="right" width="20%">姓名：</td>
<td><input type="text" name="username" pattern="^[\u4e00-\u9fa5\uf900-
\ufa2d]{1,11}$" />(汉字，只能包含中文字符(长度小于 12，大于 2))</td>
```

步骤 02 第二行第二列供用户输入腾讯 QQ 号码。代码如下：

```
<td align="right">QQ 号码：</td>
<td><input type="text" name="myqq" pattern="^[1-9][0-9]{4,}$" />(腾讯 QQ
号从 10000 开始)</td>
```

步骤 03 第三行第二列提示用户输入固定的电话号码，其形式是“XXXX-XXXXXXX”或者“XXX-XXXXXXXX”。代码如下：

```
<td align="right">固定电话：</td>
<td><input type="tel" name="mytel" pattern="\d{3}-\d{8}|\d{4}-\d{7}"/>(国
内电话号码(0511-4405222、021-87888822))</td>
```

步骤 04 继续添加供用户输入手机号码的输入框，并且要求用户输入的手机号码是以 13、14、15 或 18 开头。代码如下：

```
<td align="right">手机号码：</td>
<td><input type="text" name="myphone" pattern="^(13[0-
9]|14[5|7]|15[0|1|2|3|5|6|7|8|9]|18[0|1|2|3|5|6|7|8| 9])\d{8}$"/>(以 13、
14、15、18 开头的电话号码)</td>
```

步骤 05 添加供用户输入身份证号的输入框，并且要求输入的身份证号合法。页面代码如下：

```
<td align="right">身份证号：</td>
<td><input type="text" name="mycard"  pattern="^\d{15}|\d{18}$"/>(15 位或
18 位身份证号)</td>
```

步骤 06 添加用于执行提交操作的按钮，具体代码不再给出。

步骤 07 运行页面，查看效果，输入内容后，单击按钮进行测试，如图 7-37 所示为 Chrome 浏览器中的验证效果。

步骤 08 继续向该浏览器中的其他输入框输入内容进行测试。

步骤 09 不同的浏览器导致验证的效果有所不同，如图 7-38 所示为 Firefox 浏览器验证 QQ 号码时的效果。

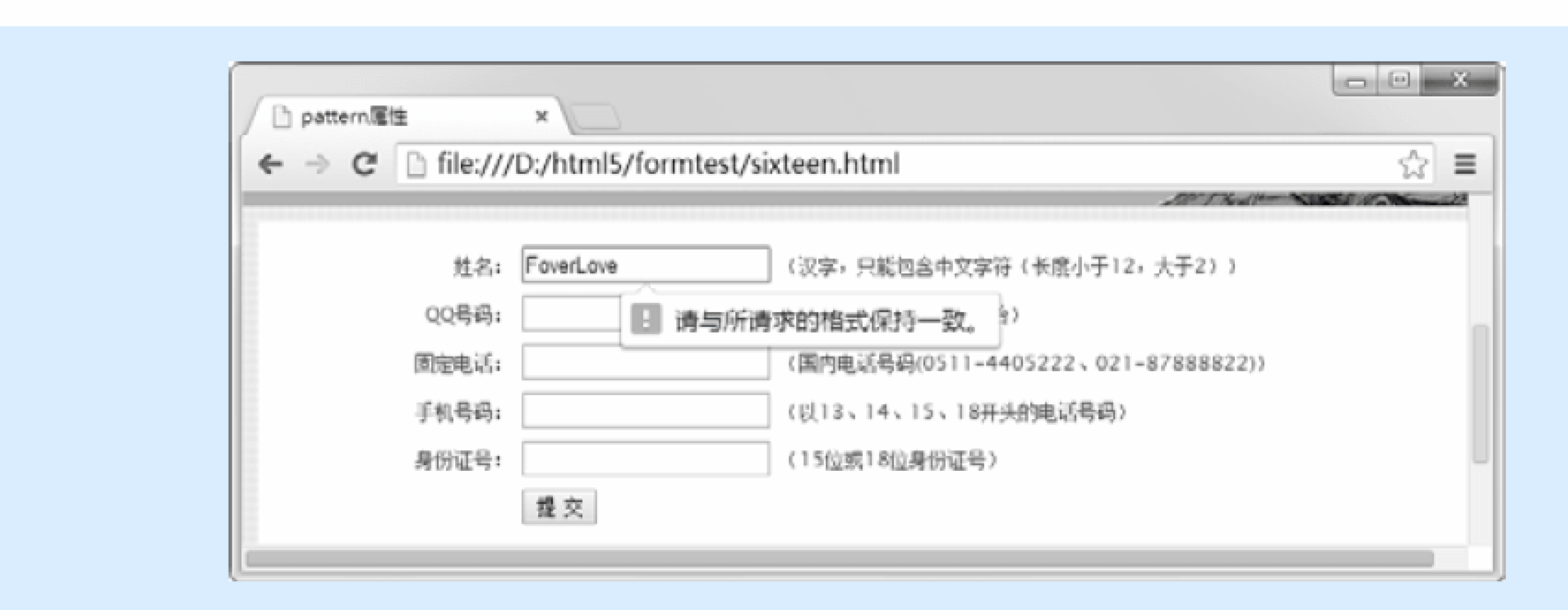


图 7-37　Chrome浏览器中的验证效果

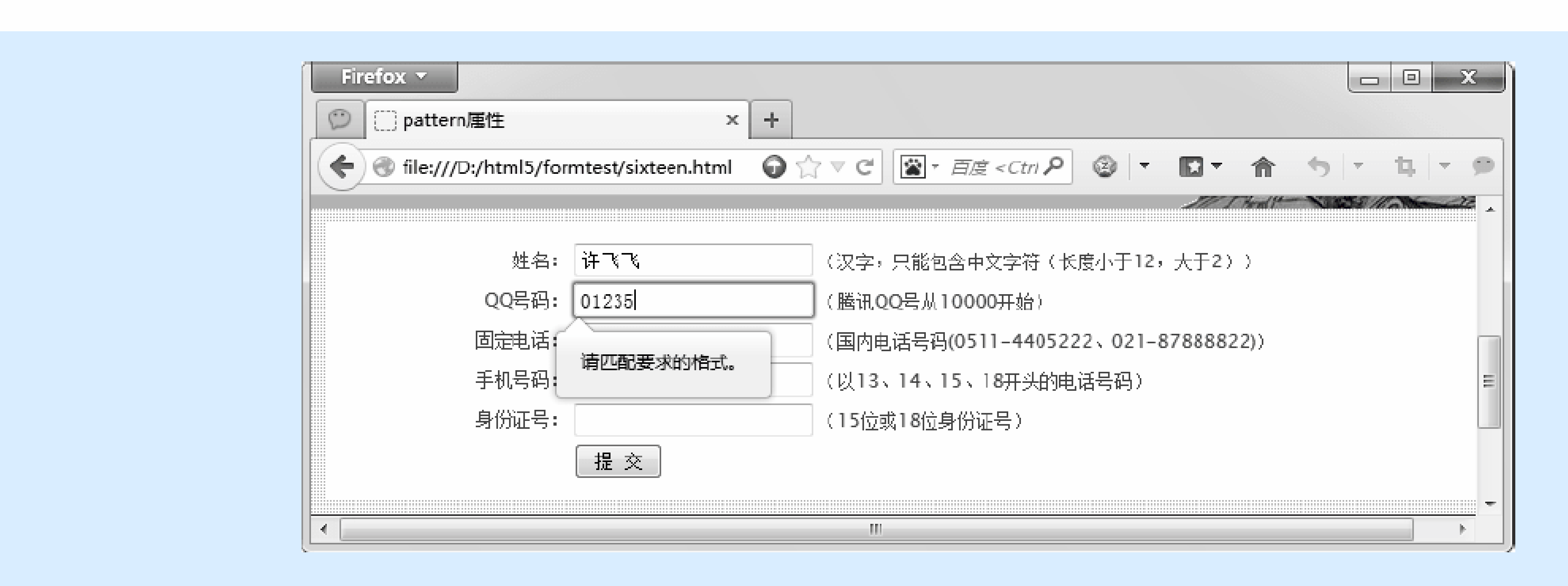


图 7-38　Firefox验证QQ号码时的效果

例 7.16 进行匹配时，“^”表示开头，“$”表示结尾。除了该示例介绍的这些表达式外，使用 pattern 属性还可以指定正则表达式匹配其他信息。例如，在表 7-5 中列出了其他常用的正则表达式。

表 7-5　常用的正则表达式和说明

<table>
<tr><th>正则表达式</th><th>说　明</th></tr>
<tr><td>^[0-9]*$</td><td>数字</td></tr>
<tr><td>^\d{n}$</td><td>n 位的数字</td></tr>
<tr><td>^\d{n,}$</td><td>至少 n 位的数字</td></tr>
<tr><td>^\d{m,n}$</td><td>m~n 位的数字</td></tr>
<tr><td>^([1-9][0-9]*)+(.[0-9]{1,2})?$</td><td>非零开头的最多带两位小数的数字</td></tr>
<tr><td>^(\-|\+)?\d+(\.\d+)?$</td><td>正数、负数和小数</td></tr>
<tr><td>^\d+$ 或 ^[1-9]\d*|0$</td><td>非负整数</td></tr>
<tr><td>^-[1-9]\d*|0$ 或 ^((-\d+)|(0+))$</td><td>非正整数</td></tr>
<tr><td>^[\u4e00-\u9fa5]{0,}$</td><td>汉字</td></tr>
<tr><td>^[A-Za-z0-9]+$ 或 ^[A-Za-z0-9]{4,40}$</td><td>英文和数字</td></tr>
<tr><td>^[A-Za-z]+$</td><td>由 26 个英文字母组成的字符串</td></tr>
<tr><td>^[A-Za-z0-9]+$</td><td>由数字和 26 个英文字母组成的字符串</td></tr>
</table>

续表

正则表达式	说　明
^\w+$ 或 ^\w{3,20}$	由数字、26 个英文字母或者下划线组成的字符串
^[\u4E00-\u9FA5A-Za-z0-9_]+$	中文、英文、数字包括下划线
^[\u4E00-\u9FA5A-Za-z0-9]+$ 或[\u4E00-\u9FA5A-Za-z0-9]{2,20}$	中文、英文、数字但不包括下划线等符号
^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$	E-mail 地址
[a-zA-z]+://[^\s]* 或^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]*)?$	URL 地址
^([0-9]){7,18}(x|X)?$ 或^\d{8,18}|[0-9x]{8,18}|[0-9X]{8,18}?$	以数字、字母 x 结尾的短身份证号
^[a-zA-Z][a-zA-Z0-9_]{4,15}$	账号是否合法(字母开头，允许 5~16 字节，允许字母数字下划线)
^[a-zA-Z]\w{5,17}$	密码(以字母开头，长度在 6~18 之间，只能包含字母、数字和下划线)

4. placeholder属性

placeholder 属性提供一种提示，描述输入框所期待的值。placeholder 属性适用于类型是 text、search、url、tel、email 以及 password 的<input>标记。

在使用 placeholder 属性时，提示会在输入框为空的时候显示出来，而在输入框获得焦点时消失。

【例 7.17】

在上个示例的基础上进行更改，分别为“姓名”和“QQ 号码”输入框指定 placeholder 属性。相关的代码如下：

```
<input type="text" name="username" pattern="^[\u4e00-\u9fa5\uf900-
  \ufa2d]{1,11}$" placeholder="例如：许飞飞或欧阳小鱼" />
<input type="text" name="myqq" pattern="^[1-9][0-9]{4,}$"
  placeholder="10000" />
```

在浏览器中运行上述代码，查看效果，如图 7-39 所示。

5. required属性

在 7.3 节介绍的 HTML 5 中新增的输入类型时，这些类型并不会自动判断用户是否在输入框中已经输入内容，只有在输入框中输入内容时才会进行判断。如果开发者需要要求某个输入框的内容是必须填写的，那么可以为<input>标记指定 required 属性。

required 属性指定必须在提交之前填写输入框，即输入框不能为空。例如，用户登录时要求必须输入用户名和密码，这时可以为它们指定 required 属性。

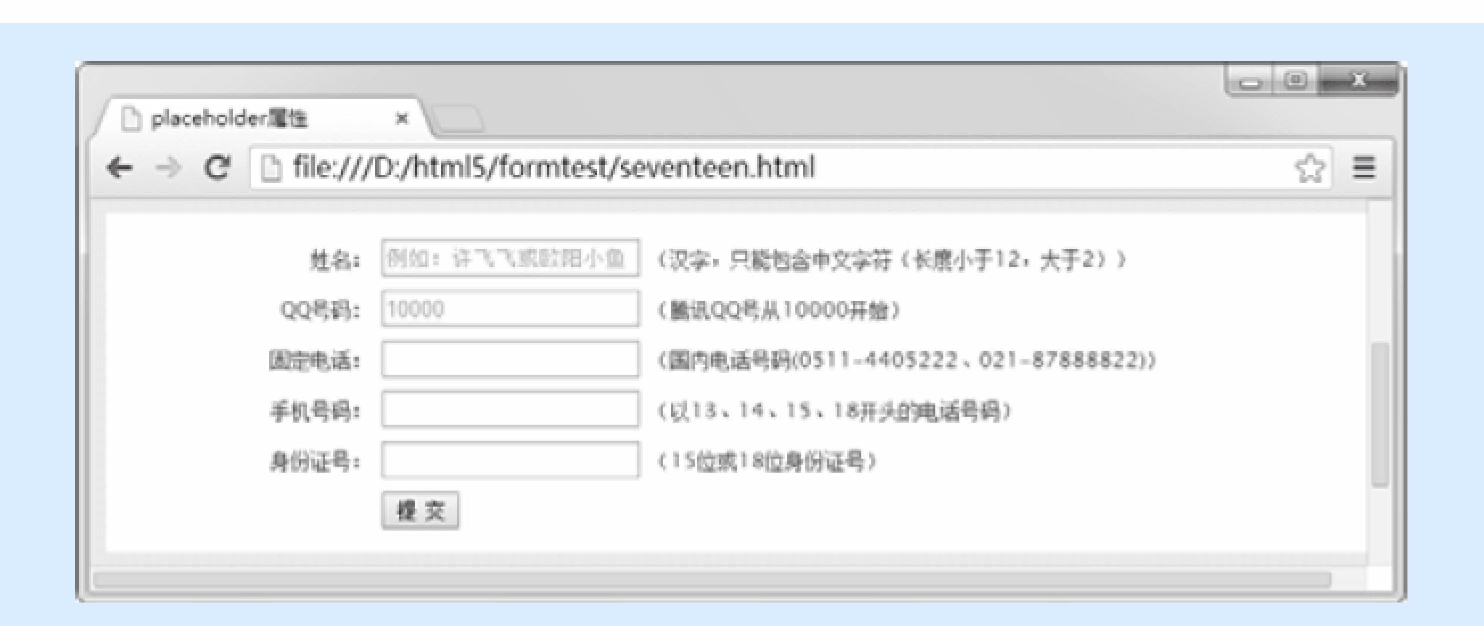


图 7-39 placeholder属性的使用

【例 7.18】

继续在上个示例的基础上添加代码，为“姓名”和“手机号码”的<input>标记指定 required 属性。相关代码如下：

```
<input type="text" name="username" pattern="^[\u4e00-\u9fa5\uf900-
  \ufa2d]{1,11}$" placeholder="例如：许飞飞或欧阳小鱼" required />
<input type="text" name="myphone" pattern="^(13[0-9]|14[5|7]|15
  [0|1|2|3|5|6|7|8|9]|18[0|1|2|3|5|6|7|8| 9])\d{8}$" required="true"/>
```

在浏览器中运行上述代码，查看效果，直接单击“提交”按钮进行测试，Chrome 浏览器中的效果如图 7-40 所示。



图 7-40 required属性的使用

浏览器不同，导致显示的效果也会有所不同，图 7-41 显示了 required 属性在 Firefox 浏览器中的效果。

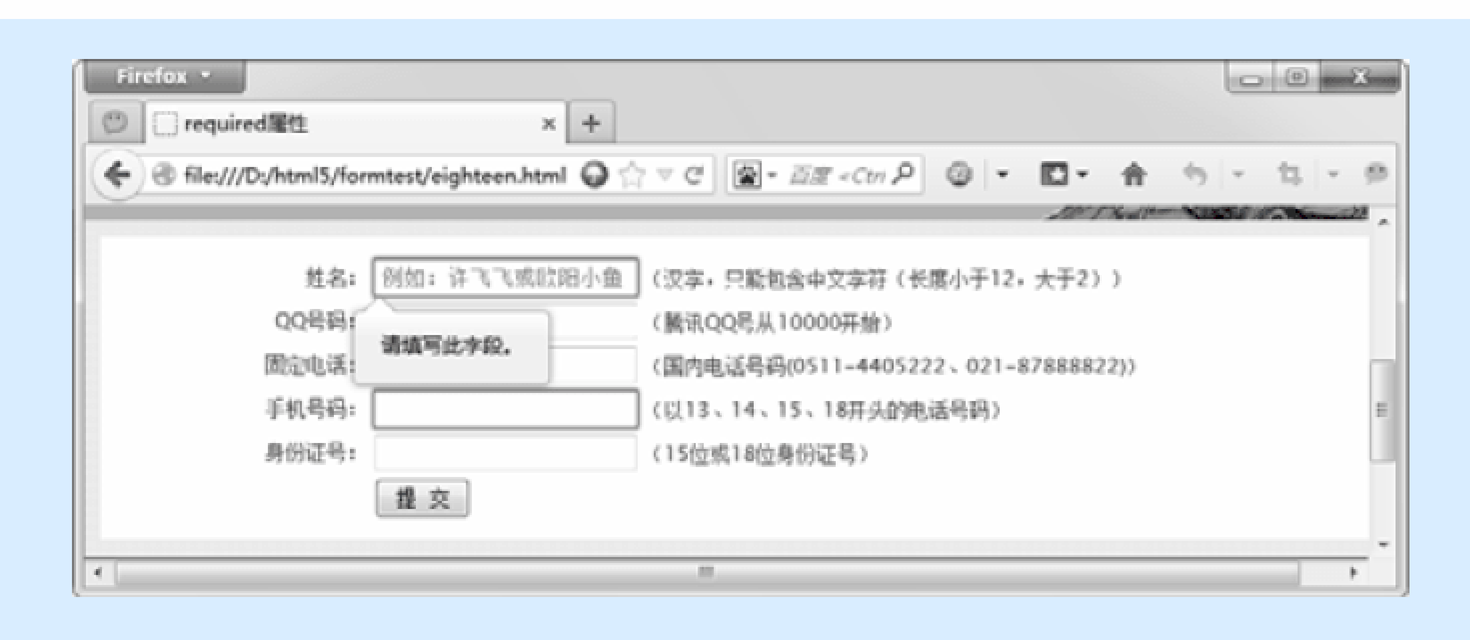


图 7-41 Firefox浏览器显示required属性的效果

7.5 实战——修改用户个人资料

目前，许多网站都提供了用户的注册信息(例如购物网站)，这些信息非常简单，包括登录账号和密码，用户注册成功后，可以在修改页面修改个人资料进行补充，本节实战将本章介绍的内容结合起来，实现用户个人资料修改页面的设计，并且根据需要实现这些信息的基本验证。

修改用户资料页面非常简单，大多数知识都是利用本章介绍的属性、类型和元素。实现步骤如下。

步骤 01 向 HTML 网页中添加表单元素，指定<form>标记的 action 属性、method 属性和 autocomplete 属性。代码如下：

```
<form action="#" method="get" autocomplete="off">
    <!-- 省略其他内容 -->
</form>
```

步骤 02 向表单元素中添加一个 10 行两列的表格，并且为表格指定标题。部分代码如下：

```
<table align="center" width="100%">
   <caption><h1>修改或者完善用户个人资料</h1></caption>
   <!-- 省略其他内容 -->
</table>
```

步骤 03 显示用户登录名信息，一般情况下，用户登录名不能更改，只能查看。相关代码如下：

```
<td align="right">用户登录名：</td>
<td><input type="text" name="loginname" value="hellome@163.com" disabled
readonly size= "50"/>(不能修改，只能查看)</td>
```

步骤 04 添加供用户输入真实姓名的输入框，通过 pattern 属性验证输入的内容只能是汉字，placeholder 属性显示一个基本提示，required 属性指定该输入框是必填的，autofocus 属性指定默认的焦点。代码如下：

```
<td align="right">真实姓名：</td>
<td><input type="text" name="realname" pattern="^[\u4e00-\u9fa5]{0,}$"
placeholder="例如：许鱼" required autofocus size="50"/>(必须填写，只能输入汉
字)</td>
```

步骤 05 添加供用户输入真实年龄的输入框，指定 type 属性的值为 number，并且年龄在 15 岁到 120 之间，该项是必填的，默认年龄是 24 岁。代码如下：

```
<td align="right">真实年龄：</td>
<td><input type="number" name="realage" value="24" min="15" max="120"
required="true" style="width:335px;"/>(必须填写)</td>
```

步骤 06 添加供用户输入或选择出生日期的输入框，将 type 属性的值设置为 date，并且该项是必填的。代码如下：

```
<td align="right">出生日期：</td>
<td><input type="date" name="birthday" value="1990-10-10"
required="required" size="50" style="width:335px;"/>(必须填写)</td>
```

步骤 07 添加供用户输入电子邮箱的输入框，允许用户输入多个电子邮箱，多个邮箱之间通过逗号进行分隔，并且该项是必填的。代码如下：

```
<td align="right">电子邮箱：</td>
<td><input type="email" name="myemail"
placeholder="12345@qq.com,sale@163.com" required multiple="true"
size="50" />(必须填写)</td>
```

步骤 08 添加供用户输入身份证号的输入框，该项是必填的。用户身份证号有 15 位或 18 位，并且身份证号的最后一位可以由数字或者字母 X 结尾，通过 pattern 属性进行验证。代码如下：

```
<td align="right">身份证号：</td>
<td><input type="text" name="card" required pattern="^\d{8,18}|[0-9x]
{8,18}|[0-9X]{8,18}?$" size="50"/>(必须填写，能够以数字或字母 x 结尾的短身份证
号)</td>
```

步骤 09 用户必须输入自己的手机号码，手机号码必须以 13、14、15 和 18 开头，其中，以 14 开头的电话号码必须是 145 和 147，以 15 和 18 开头的电话号码，其第 3 位数字不能为 4，即不能是 154 或者 184。代码如下：

```
<td align="right">手机号码：</td>
<td><input type="tel" name="card" required pattern="^(13[0-9]
|14[5|7]|15[0|1|2|3|5|6|7|8|9]|18[0|1|2|3|5|6|7| 8|9])\d{8}$"
size="50"/>(必须填写，手机号码必须以 13、14、15 和 18 开头)</td>
```

步骤 10 页面中包含个人主页，该项是选填的，并且需要设置<input>标记的 list 属性、placeholder 属性和 pattern 属性。代码如下：

```
<td align="right">个人主页：</td>
<td><input type="url" name="myaddress" list="urllist"
placeholder="http://www.baidu.com" pattern= "^htt p://([\w-]+\.)+[\w-]+
(/[\w-./?%&=]*)?$" size="50"/>(选填)</td>
<datalist id="urllist">
   <option>http://www.baidu.com</option>
   <option>http://t.qq.com</option>
   <option>http://www.sina.com</option>
</datalist>
```

步骤 11 向页面中提供一个供用户选择的幸运颜色项，该项是选填的，并且默认值为#FFBBCC。代码如下：

```
<td align="right">幸运颜色：</td>
<td><input type="color" name="lovecolor" value="#FFBBCC"/>(选填)</td>
```

步骤 12 添加供用户进行操作的“提交”按钮和“重置”按钮。代码如下：

```
<td><input type="submit" value="提 交" /> <input type="reset" />
</td>
```

步骤 13　在浏览器中运行上述代码，查看效果，图 7-42 显示了初始效果。

图 7-42　修改用户个人资料页面的初始效果

步骤 14　向页面中的输入框中输入内容进行测试，输入完毕后单击“提交”按钮提交信息，验证效果如图 7-43 所示。

图 7-43　验证用户输入的内容

7.6　表 单 验 证

HTML 5 中新增了大量的输入类型、表单属性和表单元素，同时也加强了对表单元素的验证功能。表单验证是一套系统，它为终端用户检测无效的数据并标记这些错误。下面简单了解表单验证的知识，并且介绍几种验证操作。

7.6.1 表单验证概述

表单验证是一种用户体验的优化，让 Web 应用程序更快速地抛出错误，但是它不能取代服务器端验证。前端验证是可以绕过的，因此，重要数据还要依赖于服务器端的验证。在 HTML 网页中，使用表单验证有两个好处，它不仅可以避免信息无法更新或出现新错误，而且减轻了服务器端的压力。

根据验证的提交方式，可以将验证分为自动验证、显式验证和自定义验证三种。显式验证和自定义验证分别在 7.6.2 和 7.6.3 小节中进行介绍。另外，可以取消表单验证，取消验证参考 7.4.1 小节的 novalidation 属性说明。

自动验证是 HTML 5 表单的默认验证方式，它会在表单提交时执行自动验证。如果输入框的内容不合法，那么验证将无法通过，不会提交表单。HTML 5 中新增加的多种属性都实现了自动验证的功能。

- required：该属性限制在提交时元素内容不能为空。如果元素中的内容为空，则不允许提交。
- pattern：该属性根据设置的正则表达式的格式验证输入的内容是否有效，如果无效，则不允许进行提交。
- max 和 min：这两个属性限制数值类型输入范围的最大值和最小值，不在范围内的不允许进行提交。
- step：该属性限制元素的值每次增加或者减少的基数，不是基数倍数时不允许提交。例如，当想让用户输入的值在 0~100 之间，而且必须是 5 的倍数时，可以将 step 属性的值指定为 5。

7.6.2 checkValidity()验证

自动验证可以对用户输入的内容进行验证，但是，有时候，自动验证并不能够完全满足开发者的要求，这时可以使用显式验证。显示验证需要调用 checkValidity()方法，它对表单内所有元素或者单个元素进行有效性验证。form 元素、input 元素、select 元素和 textarea 元素都具有 checkValidity()方法，该方法以布尔值的形式返回验证结果。另外，form 元素和 input 元素都存在一个 validity 属性，这个属性返回 ValidityState 对象。

【例 7.19】

通过 checkValidity()方法验证用户输入的内容是否符合要求，实现步骤如下所示。

步骤 01 向页面中添加两个输入框和一个按钮，第一个输入框表示生日，其 type 属性值为 date 类型，第二个输入框表示邮件地址，其 type 属性值为 email。代码如下：

```
<form action="test.html" method="get">
       生   日: <input id="birth" name="birth"
      type="date"/><br/>
    邮件地址: <input id="email" name="email" type="email"/><br/>
    <input type="submit" value="提交" onclick="return check();"/>
</form>
```

步骤 02　从上个步骤中可以看出，按钮具有一个 onclick 事件属性，该属性调用 check()函数。该函数的代码如下：

```
<script type="text/javascript">
    var check = function() {
        return commonCheck("birth", "生日", "字段必须是有效的日期！")
            && commonCheck("email", "邮件地址", "字段必须符合电子邮件的格式！");
    }
</script>
```

步骤 03　从步骤 02 中可以看出，check()函数直接返回调用的 commonCheck()函数的结果。commonCheck()函数包含 3 个参数：第一个参数指定字段的 id 属性值；第二个参数指定字段名称；最后一个参数指定提示信息。函数代码如下：

```
var commonCheck = function(field , fieldName , tip) {
    var targetEle = document.getElementById(field);
    if (targetEle.value.trim() == "") {             // 如果该字段的值为空
        alert(fieldName + "字段必须填写！");
        return false;
    } else if(!targetEle.checkValidity()) {         // 执行输入校验
        alert(fieldName + tip);
        return false;
    }
    return true;
}
```

在上述代码中，首先通过 document.getElementById()获取网页中指定的 id 属性值对象，接着判断该对象的 value 属性值是否为空，如果为空，弹出“**字段必须填写！”的提示；并且返回 false；如果字段不为空，则调用 checkValidity()方法执行输入验证，并弹出提示，且返回结果为 false。

步骤 04　在浏览器中运行本次示例的代码，单击按钮进行测试，查看 checkValidity()方法的验证结果，如图 7-44 所示。

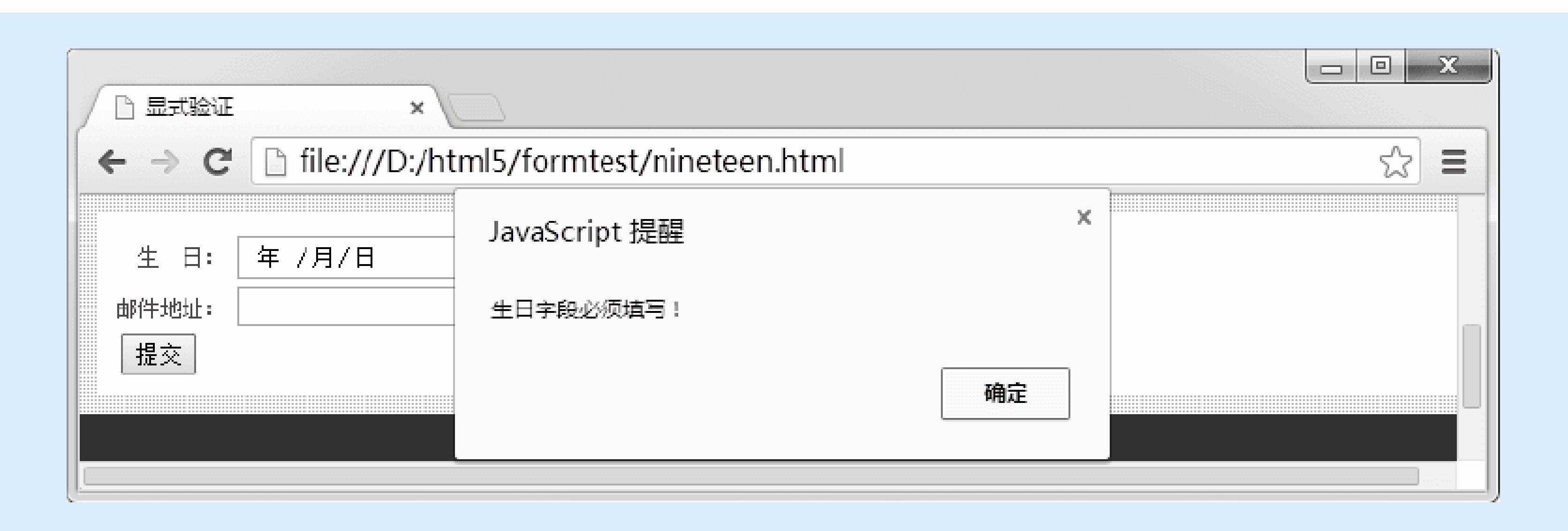


图 7-44　checkValidity()方法的验证效果

7.6.3 setCustomValidity()验证

setCustomValidity()方法适用于 HTML 5 中的所有类型的 input 元素，通过自定义的方式对用户输入的信息进行验证。setCustomValidity()方法与 checkValidity()方法一样，通常都会结合 JavaScript 脚本使用。

【例 7.20】

继续在上个示例的基础上更改内容，利用上个示例的页面，通过 setCustomValidity()方法自定义验证信息。实现步骤如下。

步骤 01 向 HTML 网页中添加输入框和按钮，并且为第二个输入框指定 required 属性。代码如下：

```
<form action="#" method="get">
      生   日：<input id="birth" name="birth"
    type="date"/><br/>
   邮件地址：<input id="email" name="email" type="email" required /><br/>
   <input type="submit" value="提交" onclick="testcheck();"/>
</form>
```

步骤 02 添加 JavaScript 脚本函数，在函数中分别判断“生日”输入框和“邮件地址”输入框的内容是否为空。函数代码如下：

```
<script type="text/javascript">
    var testcheck = function(){
        var birthEle = document.getElementById("birth");
        if (birthEle.value.trim() == "") { // 如果该字段的值为空
            birthEle.setCustomValidity("生日怎么能不填写呢？快来输入吧！");
        } else {
            birthEle.setCustomValidity("");
        }
        var emailEle = document.getElementById("email");
        if (emailEle.value.trim() == "") { // 如果该字段的值为空
            emailEle.setCustomValidity("还没有输入您的邮箱呢？请输入");
        } else {
            emailEle.setCustomValidity("");
        }
    }
</script>
```

步骤 03 在浏览器中运行上述代码，查看效果，输入内容后，单击按钮进行测试，如图 7-45 所示为提示用户输入邮件地址的效果。从该图中可以看出，虽然为“邮件地址”输入框指定了 required 属性，但是它并没有提示，而是显示了自定义的信息。



图 7-45　自定义验证 1

步骤 04 向邮件地址中输入内容后，继续单击“提交”按钮提交信息，此时会自动验证用户输入内容的合法性，效果如图 7-46 所示。



图 7-46 自定义验证 2

步骤 05 根据提示信息更改邮件地址的内容，更改以后再次单击按钮进行提交。

7.7 本 章 习 题

1. 填空题

(1) 表单由表单元素、__________和表单按钮三部分组成。

(2) __________元素是密钥生成器，作用是提供一种验证用户的可靠方法。

(3) output 元素最常用的 3 个属性是 for、form 和__________。

(4) __________类型是一种专门用于输入搜索关键词的输入框。

(5) multiple 属性适用于类型是__________和 file 的<input>标记。

(6) __________属性提供一种提示，该提示会在输入框为空时显示出来，在输入框获得焦点时消失。

2. 选择题

(1) datalist 元素与 input 元素结合使用时，需要指定 datalist 元素的__________属性和 input 元素的__________属性。

A. id、name　　B. name、id　　C. id、list　　D. list、id

(2) HTML 5 中新增的输入类型不包括__________。

A. email　　B. checkbox　　C. range　　D. number

(3) 在如图 7-47 所示的 Chrome 效果图中，一定不会使用到__________输入类型。

图 7-47 测试效果图

A. number　　B. range　　C. url　　D. email

(4) number 输入类型的输入框能够设置对所接受的数值的限定。该输入类型和其他属

性结合使用时，可以通过设置________属性指定输入域合法的间隔，该属性的默认值为 1。

A. value　　B. min　　C. max　　D. step

(5) HTML 5 中新增的与表单有关的两个属性是__________。

A. autocomplete 和 novalidate　　B. autocomplete 和 autofocus

C. autofocus 和 novalidate　　D. autocomple 和 required

(6) HTML 5 中新增的__________属性指定输入框的内容必须填写。

A. multiple　　B. autofocus　　C. pattern　　D. required

3. 上机练习

(1) 利用新增的输入类型和属性设计网页

根据图 7-48 的效果设计页面，页面中可以使用本章介绍的元素、输入类型和属性。其中，通过 datalist 元素定义了一系列的选项列表，供用户在“姓名”输入框选择，选项列表如图 7-49 所示。



图 7-48　页面效果



图 7-49　datalist元素的作用效果

(2) 表单的显式验证和自定义验证

读者可以根据图 7-48 和 7-49 的效果，通过显式验证或者自定义验证的方式验证表单中输入框的内容是否合法。在验证过程中，可以自定义验证的信息提示。





HTML 5 操作页面图形

HTML 5 规范引进了许多新特性，其中最令人期待之一的就是 canvas 元素。HTML 5 中的 canvas 提供了通过 JavaScript 绘制图形的方法，该方法使用简单，但是功能非常强大。可以通过它来动态生成和展示图形、图表、图像以及动画，每一个 canvas 元素都有一个上下文对象，在其中可以绘制任意图形。本章介绍 2D canvas 基础知识以及如何使用基本 canvas 方法绘制图形。

通过本章的学习，读者不仅可以了解 canvas 的历史与浏览器的支持情况，还可以绘制各种图形，例如线条、三角形、矩形、圆形、扇形、贝塞尔曲线、线性和径向渐变以及图片等，还可以熟练地对图形进行旋转、平移和缩放等操作。

本章学习目标：

- 了解 canvas 元素的历史
- 掌握如何绘制文本、矩形
- 熟悉与路径有关的方法
- 掌握线条、圆形、贝塞尔曲线的绘制
- 掌握图形变换效果的实现

- 熟悉 transform()方法的使用
- 掌握 drawImage()方法的使用
- 了解如何实现动画效果

8.1　了解 canvas 元素

canvas 元素是 HTML 5 中新增加的一个元素，专门用来绘制图像、图形、文字及动画等。本节简单了解 canvas 元素的基础知识，包括它的历史、概念、与 CSS 的关系以及 Canvas API 等内容。

8.1.1　canvas 历史

canvas 的概念最初是由苹果公司提出的，用于在 Mac OS X WebKit 中创建控制板部件(Dashboard Widget)。在 canvas 出现之前，Web 开发者如果要在浏览器中使用绘图 API，只能使用 Adobe 的 Flash 和 SVG(Scalable Vector Graphics，可伸缩矢量图形)插件，或者只有 IE 才支持的 VML(Vector Markup Language，矢量标记语言)，以及其他一些稀奇古怪的 JavaScript 技巧。

假设 Web 开发者要在没有 canvas 元素的条件下绘制一条对角线，这听起来非常简单，但实际上如果没有一套二维绘图 API 的话，将会是一项相当复杂的工作。在 HTML 5 中，canvas 能够提供这样的功能，对浏览器端来说此功能非常有用，因此，canvas 被纳入了 HTML 5 规范。

最初，苹果公司曾暗示可能会为 WHATWG 草案中的 canvas 规范申请知识产权，这在当时引起了 Web 标准化追随者的关注，不过，苹果公司最终还是按照 W3C 的免版税专利权许可条款公开了其专利。

8.1.2　canvas 元素

canvas 元素只是一个图形的容器，它本身不具有任何行为，但是它把一个 API 展现给客户端脚本，所有使用者必须将它与脚本结合起来绘制图形，以便脚本能够把想绘制的东西都绘制到一块画布上。

在网页上使用 canvas 元素时，它会创建一块矩形区域。如下是 HTML 页面中最基本的 canvas 元素：

```
<canvas></canvas>
```

在默认情况下，canvas 元素创建的矩形区域宽度为 300 像素，高度为 150 像素，但 Web 开发者也可以自定义具体的大小或者设置 canvas 元素的其他属性。代码如下：

```
<canvas width=200 height=200 id="djx" style="border:1px solid red;"> </canvas>
```

尽管 canvas 元素的功能非常强大，用处也很多，但是在某些情况下，如果其他元素已经够用了，就不应该再使用 canvas 元素。例如，使用 canvas 元素在 HTML 页面中动态绘制所有不同的标题，还不如直接使用标题样式元素(例如 h1、h2 和 h3 等)，它们的实现效果是一样的。

8.1.3 CSS 和 canvas

同大多数 HTML 元素一样，canvas 元素也可以通过应用 CSS 的方式来增加边框，设置内边距、外边距等，而且一些 CSS 属性还可以被 canvas 内的元素继承。例如字体样式，在 canvas 内添加的文字，其样式默认同 canvas 元素本身是一样的。

另外，在 canvas 中为 context 设置属性同样要遵从 CSS 语法。例如，对 context 应用颜色和字体样式，这跟在任何 HTML 和 CSS 文档中使用的语法完全一样。

8.1.4 Canvas API

canvas 元素只是一个容器，它并不能够做任何事情，在 Windows 中绘制图形时需要得到一个上下文设备 DC，同样，使用 canvas 元素绘图时，需要先得到一个渲染上下文对象 Context。

上下文对象可以让各种不同的图形设备在外界看起来都是同一个样式，这样读者只需要关注绘图，其他的工作都可以交给操作系统和浏览器。简单地说，上下文对象就是把各式各样的“具体”变成一个统一的“抽象”，从而减轻开发者的负担。

使用 canvas 绘图时并不是直接绘制到屏幕上的，而是先画到一个上下文 Context 上，然后再刷新到屏幕上。

每个 canvas 元素都有一个对应的 Context 对象，并且该对象是唯一的。Canvas API 定义在这个 Context 对象上，因此需要获取 Context 对象。获取代码如下：

```
var canvas = document.getElementById("myCanvas");
if(canvas.getContext){
    var ctx = canvas.getContext("2d");
}
```

在上述代码中，getContext()方法指定一个参数“2d”，表示该 canvas 对象用于生成 2D 图案，即平面图案。如果是参数“3d”，就表示用于生成 3D 图像(即立体图案)，这部分实际上单独叫作 WebGL API(本章不涉及)。

提示：上下文对象中提供了用于在画布上绘图的方法和属性，这些属性和方法用于在画布上绘制文本、线条、矩形和圆形等。关于这些属性和方法，会在后面的小节中绘制图形时进行介绍。

8.1.5 浏览器支持情况

目前，许多主流的浏览器都提供了对 canvas 元素的支持，用户可以从测试网站上查看浏览器对该元素的支持情况，如图 8-1 所示。

在使用 canvas 元素之前，可以通过代码判断当前使用的浏览器是否支持该元素。一般情况下，用以下 3 种方法进行判断。

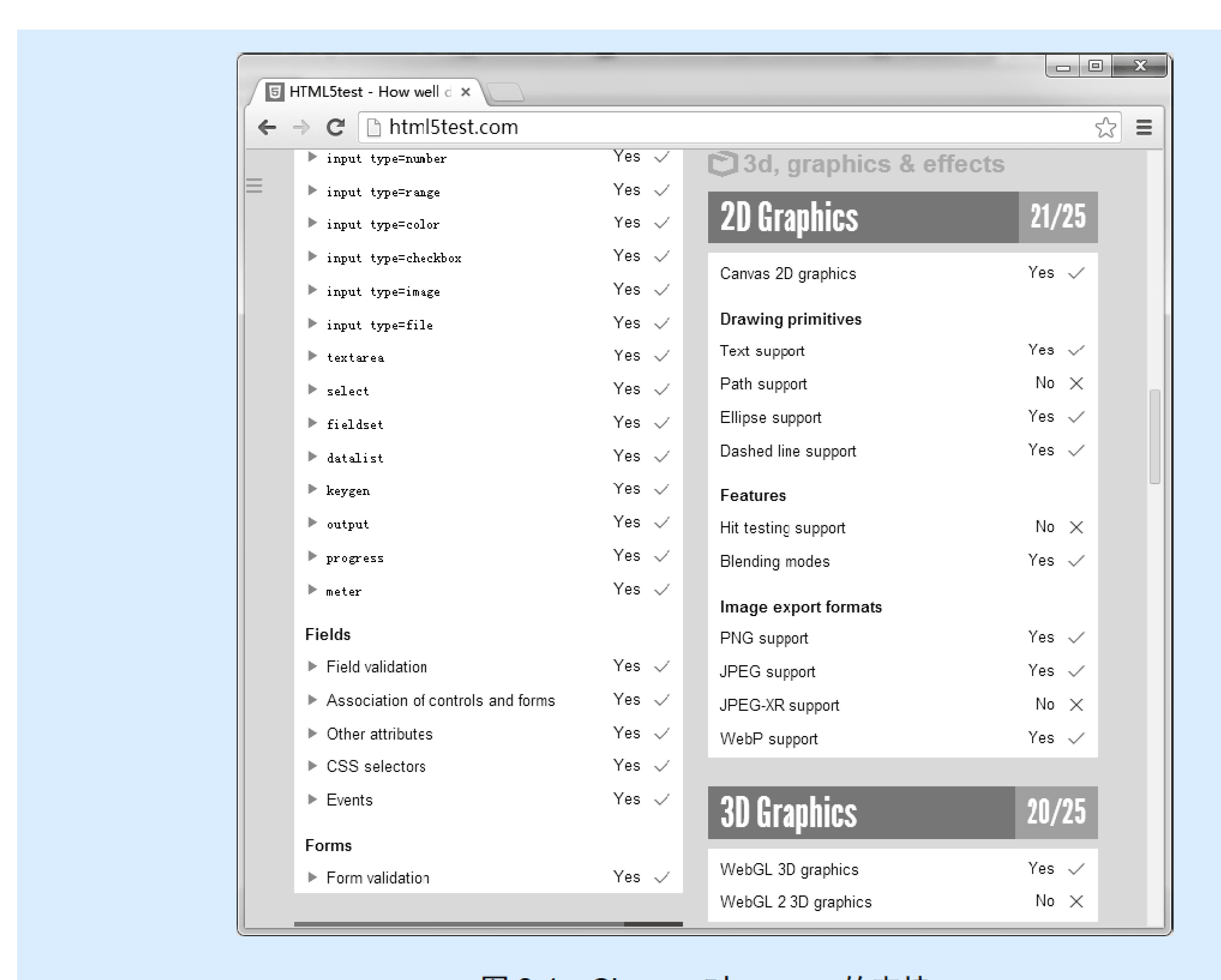


图 8-1　Chrome 对 canvas 的支持

1．向元素的开始和结束标记添加内容

canvas 元素与 audio 和 video 等元素一样，可以直接在该元素的开始标记和结束标记之间添加代码，如果浏览器不支持该元素，则会显示标记之间的内容。代码如下：

```
<canvas>
    当前浏览器不支持 canvas 元素。
</canvas>
```

2．判断 getContext()方法

可以直接在脚本中判断元素的 getContext()方法是否存在，判断代码如下：

```
var canvas = document.getElementById("myCanvas");
if (canvas.getContext) {
    alert("您的浏览器不支持 canvas 元素");
} else {
    alert("您的浏览器支持 canvas 元素");
}
```

3．try-catch 语句

在脚本中使用 try-catch 语句也可以判断浏览器对 canvas 元素的支持情况，try 语句块

中直接获取上下文对象，获取过程中出现异常时，直接抛出。代码如下：

```
try {
    document.createElement("myCanvas").getContext("2d");
    document.getElementById("support").innerHTML= "您的浏览器支持 canvas 元素";
} catch (e) {
    document.getElementById("support").innerHTML = "您的浏览器不支持 canvas 元素";
}
```

8.2 绘制文本

canvas 画布提供了一个用来作图的平面空间，该空间的每个点都有自己的坐标，x 表示横坐标，y 表示纵坐标。原点(0，0)位于图像左上角，x 轴的正向是原点向右，y 轴的正向是原点向下。

无论是绘制文本，还是绘制其他图形，如果不指定坐标，就都从坐标原点(0，0)开始绘制。文本是用户最经常见到的内容，通过上下文对象提供的属性和方法可以绘制出指定的文本信息，本节将向读者介绍如何绘制文本。

8.2.1 绘制普通文本

绘制文本时需要使用到上下文对象中的属性和方法，通过属性可以设置文本的字体样式和对齐方式等信息，常用的 3 个属性说明如下所示。

- font：设置或返回文本内容的当前字体。
- textAlign：设置或返回文本内容的当前对齐方式，其属性值可以是 start(默认值)、end、right 和 center。
- textBaseline：设置或返回在绘制文本时使用的当前文本基线，其属性值可以是 top、hanging、middle、alphabetic、ideographic(默认值)和 bottom。

Web 开发者可以通过 3 种方法绘制文本，这 3 种方法的说明如下所示。

- fillText()：该方法在画布上绘制“被填充的”文本。
- strokeText()：该方法在画布上绘制文本(无填充)。
- measureText()：该方法返回包含指定文本宽度的对象。

fillText()方法在画布上绘制“被填充的”文本，而 strokeText()方法直接在画布上绘制无填充的文本。这两个方法都包含 4 个参数，参数说明都一样。以 fillText()方法为例，基本语法如下：

```
context.fillText(text, x, y, maxWidth);
```

在上述语法中，text 参数指定在画布上输出的文本；x 表示开始绘制文本的 x 坐标位置(相对于画布)；y 表示开始绘制文本的 y 坐标位置(相对于画布)；maxWidth 是一个可选参数，指定允许的最大文本宽度，单位是像素。

measureText()方法返回一个对象，该对象包含以像素指定的字体宽度。如果需要在文本向画布输出之前就了解文本的宽度，那么可以使用该方法。该方法的语法如下：

```
context.measureText(text).width;
```

在上述语法中，使用 measureText()方法时需要传入一个 text 参数，该参数表示要测量的文本。

【例 8.1】

向 HTML 网页中添加代码显示一首古诗，标题通过 strokeText()方法绘制，内容则通过 fillText()方法进行绘制。完整的实现步骤如下。

步骤 01 在 HTML 页面的合适位置添加 canvas 元素，并指定其宽度、高度和唯一标识 ID 属性。代码如下：

```
<canvas id="MyCanvas" width="720" height="300">当前浏览器不支持 canvas 元素
</canvas>
```

步骤 02 页面加载时绘制古诗标题、作者和内容，通过 JavaScript 代码为页面指定 load 事件，首先绘制古诗标题。代码如下：

```
window.onload = function(){
    var title = "将进酒";
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");          //获取上下文对象
        context.font = "bold 30px sans-serif";          //设置字体样式
        context.strokeStyle = "red";                    //设置笔触的颜色
        context.strokeText(title,250,30);               //绘制无填充文本
        //省略其他绘制内容
    }
}
```

步骤 03 继续向上个步骤的脚本代码中添加新代码，指定古诗作者，通过 fillText()方法绘制作者文本，并且重新指定字体样式、填充颜色和文本基线等内容。代码如下：

```
context.font = "italic 18px sans-serif";
context.fillStyle = "blue";
context.textBaseline = "bottom";
context.fillText("(作者：李白)",340,30);
```

步骤 04 继续向前面的代码后添加绘制古诗内容的文本，指定文本的大小是 18 像素，字体样式是“隶书”，填充颜色是#404040。代码如下：

```
context.font = "18px 隶书";
context.fillStyle = "#404040";
context.fillText("君不见，黄河之水天上来，奔流到海不复回。",160,70);
context.fillText("君不见，高堂明镜悲白发，朝如青丝暮成雪。",160,90);
context.fillText("人生得意须尽欢，莫使金樽空对月。",160,110);
context.fillText("天生我材必有用，千金散尽还复来。",160,130);
context.fillText("烹羊宰牛且为乐，会须一饮三百杯。",160,150);
context.fillText("岑夫子，丹丘生，将进酒，杯莫停。",160,170);
context.fillText("与君歌一曲，请君为我倾耳听。",160,190);
context.fillText("钟鼓馔玉不足贵，但愿长醉不复醒。",160,210);
context.fillText("古来圣贤皆寂寞，惟有饮者留其名。",160,230);
context.fillText("陈王昔时宴平乐，斗酒十千恣欢谑。",160,250);
```

```
context.fillText("主人何为言少钱，径须沽取对君酌。",160,270);
context.fillText("五花马，千金裘，呼儿将出换美酒，与尔同销万古愁。",160,290);
```

步骤 05 运行上述代码，查看绘制文本的效果，如图 8-2 所示。



图 8-2　绘制文本

在例 8.1 中使用到了 fillStyle 属性和 strokeStyle 属性，这两个属性在后面小节中会经常被使用。fillStyle 属性设置或返回用于填充绘画的颜色、渐变或模式，属性的默认值是#000000。基本语法如下：

```
context.fillStyle = color | gradient | pattern;
```

从上述代码中可以看出，fillStyle 属性的值可以有 color、gradient 和 pattern 三种。

- color：指定绘图填充色的 CSS 颜色值，默认值是#000000。其值可以是十六进制颜色，也可以是颜色合法的英文名称。
- gradient：用于填充绘图的渐变对象(线性或者径向)。
- pattern：用于填充绘图的 pattern 对象。

strokeStyle 属性设置或返回用于笔触的颜色、渐变或者模式，其属性值是#000000。该属性的取值有 3 种，这些值的意义与 fillStyle 属性值的意义一样。基本语法如下：

```
context.strokeStyle = color | gradient | pattern;
```

8.2.2 绘制阴影文本

上下文对象提供了一系列与阴影有关的属性，通过这些属性，不仅可以绘制文本的阴影效果，还可以绘制图形(例如圆形和扇形)的阴影效果。例如，表 8-1 列出了上下文对象提供的阴影属性。

在表 8-1 列出的属性中，shadowOffsetX 和 shadowOffsetY 用于设置在 x 和 y 轴的延伸距离，它们不受变换矩阵的影响。为这两个属性设为负值时，表示阴影向上或向左延伸，正值则表示向下或向右延伸，它们的默认值都为 1。

【例 8.2】

在上个示例的基础上添加阴影属性，分别绘制古诗标题、古诗作者和古诗内容的文本

阴影。

表 8-1 上下文对象提供的阴影属性

属性名称	说 明
shadowColor	设置或返回用于阴影的颜色。默认值为全透明的黑色，它的值可以是标准的 CSS 颜色值
shadowBlur	设置或返回用于阴影的模糊级别，默认值为 1。其属性值必须为比 0 大的数字，它的值一般在 0~10 之间，否则将会被忽略
shadowOffsetX	设置或返回阴影距图形的水平距离。也可以理解为阴影与图形的横向位移量
shadowOffsetY	设置或返回阴影距图形的垂直距离。也可以理解为阴影与图形的纵向位移量

实现步骤如下。

步骤 01 在 JavaScript 脚本中绘制标题文本，在绘制之前分别设置阴影颜色、模糊级别以及横向和纵向位移量。代码如下：

```
context.font = "bold 30px sans-serif";
context.strokeStyle = "red";
context.shadowColor = "blue";                  //阴影颜色
context.shadowBlur = 1;                        //阴影模糊级别，这里指定为 1
context.shadowOffsetX = 2;                     //横向位移量 2
context.shadowOffsetY = -2;                    //纵向位移量-2
context.strokeText(title,250,30);                  //绘制标题文本
```

步骤 02 找到绘制古诗作者时的文本，并且分别指定 shadowColor、shadowBlur、shadowOffsetX 和 shadowOffsetY 的属性值。部分代码如下：

```
context.shadowColor = "#EE1289";               //设置阴影颜色
context.shadowBlur = 2;                        //阴影模糊级别，这里指定为 2
context.shadowOffsetX = -2;                    //横向位移量-2
context.shadowOffsetY = 2;                     //纵向位移量 2
context.fillText("(作者：李白)",340,30);
```

步骤 03 在绘制古诗作者之后、古诗内容之前重新指定阴影效果。部分内容如下：

```
context.shadowColor = "#43CD80";               //设置阴影颜色
context.shadowBlur = 3;                        //阴影模糊级别，这里指定为 3
context.shadowOffsetX = -2;                    //横向位移量-2
context.shadowOffsetY = -2;                    //纵向位移量-2
context.fillText("君不见，黄河之水天上来，奔流到海不复回。",160,70);
```

步骤 04 在浏览器中运行上述代码，查看效果，文本的阴影效果如图 8-3 所示。



图 8-3　绘制阴影文本

8.3 绘 制 矩 形

上下文对象提供了一系列绘制矩形的方法，通过这些方法，可以绘制不同的矩形，下面将介绍这些方法，并且分别通过这些方法来绘制矩形。

8.3.1 用 rect()方法绘制

调用上下文对象的方法绘制矩形时涉及到 4 个方法，它们分别是 rect()方法、fillRect()方法、strokeRect()方法以及 clearRect()方法。下面，首先对 rect()方法进行介绍。

rect()方法用于创建矩形，创建完毕后需要调用 stroke()方法或者 fill()方法在画布上实际地绘制图形。简单地说，rect()绘制的矩形不会在画布上显示，如果要显示，则需要调用 stroke()方法或者 fill()方法。rect()方法的基本语法如下：

```
context.rect(x,y,width,height);
```

在上述方法中，rect()方法需要传入 4 个参数：x 参数表示矩形左上角的 x 坐标；y 参数表示矩形左上角的 y 坐标；width 参数指定矩形的宽度，height 参数指定矩形的高度，这两个参数的单位是像素。

【例 8.3】

在例 8.2 代码的基础上添加新的代码，通过 rect()方法绘制矩形，并且分别通过调用 stroke()方法和 fill()方法在画布上显示。找到绘制文本时的 JavaScript 脚本代码，在创建上下文对象之后，调用 rect()方法创建一个宽度为 720 像素、高度为 300 像素的矩形，然后再调用 stroke()方法绘制矩形。部分代码如下：

```
window.onload = function(){
    var title = "将进酒";
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");          //获取上下文对象
        context.rect(0,0,720,300);                      //绘制矩形
        context.stroke();                //通过 stroke()方法绘制已定义的路径
        //省略其他内容
    }
}
```

在浏览器中运行上述代码，查看效果，如图 8-4 所示。



图 8-4　用 stroke()方法向画布绘制矩形

重新更改上述代码，通过 rect()方法绘制矩形完毕后，通过 fill()方法填充当前绘图路径，默认情况下填充颜色为黑色，如果要更改填充颜色，则需要通过 fillStyle 属性。更改代码如下：

```
context.rect(0,0,720,300);
context.fillStyle = "#FAF0E6";
context.fill();
```

重新刷新网页，查看效果，fill()方法绘制矩形的效果如图 8-5 所示。



图 8-5　用 fill()方法向画布绘制矩形

8.3.2　用 fillRect()方法绘制

fillRect()方法绘制“被填充”的矩形，默认的填充颜色是黑色。开发者可以使用 fillStyle 属性来设置用于绘图的颜色、渐变或者模式。fillRect()方法的基本语法如下：

```
context.fillRect(x,y,width,height);
```

在上述语法中，fillRect()方法绘制矩形时需要传入 4 个参数。x 参数绘制矩形左上角的 x 坐标；y 参数绘制矩形左上角的 y 坐标；width 和 height 则分别表示矩形的宽度和高度，以像素为单位。

【例 8.4】

fillRect()方法的效果与 rect()绘图完毕后通过 fill()方法填充时的效果一致。例如，通过 fillRect()方法显示图 8-5 中的效果。代码如下：

```
window.onload = function(){
    var title = "将进酒";
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.fillStyle = "#FAF0E6";          //填充颜色
        context.fillRect(0,0,720,300);          //直接绘图
        //省略其他内容
    }
}
```

在浏览器中重新运行本示例的代码，查看效果。

8.3.3 用 strokeRect()方法绘制

strokeRect()方法用于绘制无填充矩形，绘制时笔触的默认颜色为黑色。如果要更改颜色，需要通过 strokeStyle 属性进行设置。strokeRect()方法的基本语法如下：

```
context.strokeRect(x,y,width,height);
```

从上述语法中可以看出，strokeRect()方法的参数与 fillRect()方法一样，其含义也一样。x 和 y 分别表示矩形左上角的 x 坐标和 y 坐标，width 和 height 则分别表示矩形的宽度和高度，单位是像素。

【例 8.5】

strokeRect()方法的效果与 rect()绘图完毕后通过 stroke()方法填充时的效果一致。下面更改例 8.4 的代码，通过 strokeRect()方法绘制一个矩形，并且指定矩形的笔触颜色为#FF00FF。代码如下：

```
window.onload = function(){
    var title = "将进酒";
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.fillStyle = "#FAF0E6";          //填充颜色
        context.strokeRect(0,0,720,300);        //直接绘图
        //省略其他内容
    }
}
```

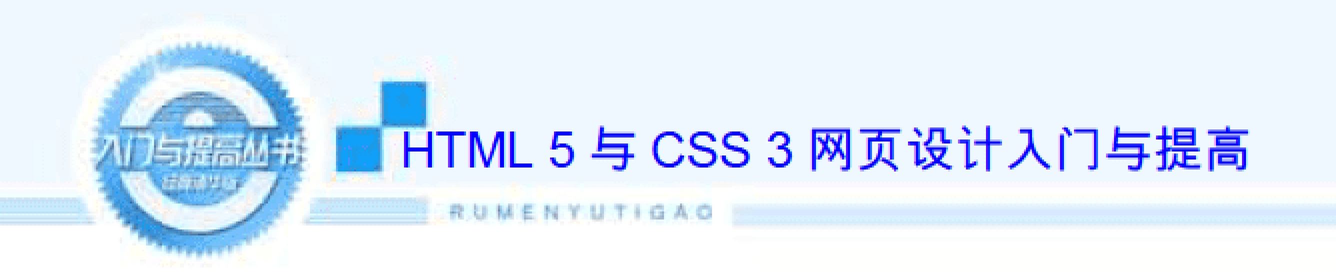
在浏览器中运行本例的代码，观察效果，绘制的矩形如图 8-6 所示。



图 8-6　用 strokeRect()方法绘制矩形

8.3.4 用 clearRect()方法清除

上下文对象提供了 clearRect()方法，用来清空指定矩形内的指定像素。使用 clearRect()方法时需要传入 4 个参数：x 和 y 分别表示要清除的矩形左上角的 x 坐标和 y 坐标；width 和 height 表示要清除矩形的宽度和高度，以像素为单位。



基本语法如下：

```
context.clearRect(x,y,width,height);
```

【例 8.6】

继续在前面的示例中添加代码，首先通过 fillRect()方法绘制一个矩形，然后清除该矩形内(150,50)坐标处长度为 400、高度为 100 的矩形。部分代码如下：

```
context.fillStyle = "#FFEFDB";      //填充颜色
context.fillRect(0,0,720,300);      //绘制矩形
context.clearRect(150,50,400,100); //从(120，50)坐标处清除长度为 400、高度为
100 的矩形
```

在浏览器中运行本次示例的代码，查看效果，使用 clearRect()方法的清空效果如图 8-7 所示。



图 8-7　clearRect()方法的使用

8.4　绘 制 路 径

上下文对象提供了一系列与路径有关的方法，使用这些方法可以创建复杂的形状和线条。在前面示例中使用到的 fill()方法和 stroke()方法都与路径有关，除了这些方法外，还有其他的一些路径方法。

下面将使用路径有关的方法绘制图形，包括简单的线条(例如线段、三角形)和复杂的形状(例如不规则的多边形)等内容。

8.4.1　路径绘图方法

在绘图之前，首先了解一下常用的路径绘图方法，这些方法及其说明如表 8-2 所示。

在表 8-2 中，绘制路径图形开始时会调用 beginPath()方法创建路径，绘制结束时调用 closePath()方法关闭路径。如果画完前面的路径没有重新指定 beginPath()方法，那么画其他路径的时候会将前面最近指定的 beginPath()后的全部路径重新绘制。另外，每次调用 context.fill()方法时会自动把当次绘制的路径的开始点和结束点相连，接着填充封闭部分。

表 8-2　上下文对象提供的路径绘图方法

方法名称	说　明
fill()	填充当前绘图路径
stroke()	绘制已定义的路径
beginPath()	起始一条路径，或者重置当前路径
moveTo()	把路径移动到画布中的指定点，不创建线条
closePath()	创建从当前点回到起始点的路径
lineTo()	添加一个新点，然后在画布中创建从该点到最后指定点的线条
clip()	从原始画布剪切任意形状和尺寸的区域
quadraticCurveTo()	创建二次贝塞尔曲线
bezierCurveTo()	创建三次方贝塞尔曲线
arc()	创建弧/曲线(用于创建圆形或部分圆)
arcTo()	创建两切线之间的弧/曲线
isPointInPath()	如果指定的点位于当前路径中，则返回 true，否则返回 false

上下文对象在调用表 8-2 中的方法绘图时，还可能使用到一些线条样式属性，这些属性及其说明如表 8-3 所示。

表 8-3　线条样式属性

属性名称	说　明
lineCap	设置或者返回线条的结束端点样式，其属性值有 3 个，说明如下。 butt：默认值，向线条的每个末端添加平直的边缘。 round：向线条的每个末端添加圆形线帽。 square：向线条的每个末端添加正方形线帽
lineJoin	设置或者返回两条线相交时所创建的拐角类型，其属性值有 3 个，说明如下。 bevel：创建斜角。 round：创建圆角。 miter：默认值，创建尖角
lineWidth	设置或返回当前的线条宽度。单位是像素
miterLimit	设置或返回最大斜接长度。斜接长度是指在两条线交汇处内角和外角之间的距离。只有当 lineJoin 属性的值为 miter 时，miterLimit 才有效。边角的角度越小，斜接长度就会越大。如果斜接长度超过 miterLimit 的值，边角会以 lineJoin 的 bevel 类型显示

8.4.2 绘制基本图形

本节的基本图形是指线段和三角形，这两种图形的实现都很简单，主要涉及到 moveTo()方法和 lineTo()方法。moveTo()方法在使用时需要传入两个参数：第一个参数表

示路径的目标位置的 x 坐标；第二个参数表示路径的目标位置的 y 坐标。基本语法如下：

```
context.moveTo(x,y);
```

lineTo()方法添加一个新点，然后创建从该点到画布中最后指定点的线条。需要注意的是：该方法并不会创建线条。使用 lineTo()方法时也需要传入两个参数，这两个参数的意义与 moveTo()方法一致。基本语法如下：

```
context.lineTo(x,y);
```

【例 8.7】

本例通过 moveTo()方法和 lineTo()方法绘制了一条宽度为 3 的线段。代码如下：

```
var canvas = document.getElementById("MyCanvas");
var context = canvas.getContext("2d");
context.beginPath();                                  //开始绘图路径
context.moveTo(30,30);                                //起始位置
context.lineTo(200,30);                               //目标坐标
context.lineWidth = 3;                                //线条宽度
context.closePath();                                  //关闭路径
context.stroke();
```

可以在一个画布中绘制多个线条，在绘制时，需要重新通过 beginPath()方法指定绘图路径。例如，重新在上个代码的基础上添加以下代码，这段代码表示重新绘制一条宽度为 5 的线段：

```
context.beginPath();
context.strokeStyle = "red";
context.lineWidth = 5;
context.moveTo(100,100);
context.lineTo(150,150);
context.closePath();
context.stroke();
```

除了绘制基本的线条外，还可以通过多个 lineTo()方法绘制三角形、矩形和多边形等。其中，三角形最常被绘制，矩形可以通过 rect()等方法绘制。

【例 8.8】

在本例中绘制两个三角形，第一个三角形通过 stroke()方法绘制，第二个三角形通过 fill()方法绘制，填充颜色为黄色，并为该图形添加阴影效果。实现步骤如下。

步骤 01 向 HTML 网页中添加 canvas 元素，并且指定该元素的宽度为 720 像素，高度为 150 像素。

步骤 02 向 JavaScript 脚本中添加代码，首先创建第一个三角形，指定宽度为 3 像素，笔触颜色为蓝色。代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.beginPath();                              //开始绘图路径
        context.lineWidth = 3;
        context.strokeStyle = "Blue";
```

```
        context.moveTo(30,30);                          //起始位置
        context.lineTo(150,30);                         //目标坐标
        context.lineTo(250,100);                        //目标坐标
        context.closePath();                            //关闭路径
        context.stroke();                               //绘图
        //省略其他内容
    }
}
```

步骤 03 在上述代码的基础上继续添加代码，指定填充颜色为蓝色，阴影效果为蓝色，模糊路径为 10。相关代码如下：

```
context.beginPath();                      //开始绘图
context.fillStyle = "yellow";             //填充颜色
context.shadowColor = "blue";
context.shadowBlur = 10;
context.shadowOffsetX= -3;
context.shadowOffsetY = -3;
context.moveTo(350,30);                 //起始坐标
context.lineTo(350,100);                //目标坐标
context.lineTo(600,75);
context.closePath();                    //关闭路径
context.fill();
```

步骤 04 在浏览器中运行上述代码，查看效果，如图 8-8 所示。



图 8-8 绘制三角形

8.4.3 绘制圆形和扇形

绘制圆形和扇形时，使用 arc()方法创建弧/曲线，或者圆形和扇形。创建圆形时，需要把起始角设置为 0，结束角设置为 2*Math.PI。arc()方法的基本语法如下：

```
context.arc(x,y,r,sAngle,eAngle,counterclockwise);
```

在使用 arc()方法时需要传入多个参数：x 和 y 表示圆的中心的 x 坐标和 y 坐标；r 表示圆的半径；sAngle 和 eAngle 分别表示以弧度为单位的起始角和结束角；counterclockwise 是一个可选参数，指定应该逆时针绘图还是顺时针绘图。将它的值设置为 false(默认值)时表示顺时针绘图，设置为 true 时表示逆时针。

简单地说，使用 arc()方法绘图时，x 和 y 决定中心角的位置，sAngle 决定起始角的位置，eAngle 决定结束角的位置。

【例 8.9】

本例通过 arc()方法创建弧、圆形和扇形等图形，并且通过 stroke()方法和 fill()向画布绘制。实现步骤如下。

步骤 01 向 HTML 网页中添加长度为 720 像素、高度为 350 像素的 canvas 元素。

步骤 02 向 JavaScript 脚本中添加代码，首先通过 arc()方法顺时针方向绘图，坐标原点是(100，100)，半径是 100，起始点为 0，结束点为 60。代码如下：

```
<script>
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.beginPath();                                    //开始绘图路径
        context.arc(100,100,50,0,Math.PI/3);
        context.closePath();                                    //关闭路径
        context.stroke();                                       //绘图
        //省略其他代码
    }
}
</script>
```

步骤 03 在上个步骤的基础上继续添加脚本代码，把用 arc()方法绘图时指定的坐标原点更改为(300,50)，半径、起始角、结束角不发生改变，但是要求逆时针方向绘图，这里不给出代码。

步骤 04 添加脚本代码，通过 fill()方法向画布绘图，arc()方法绘图时坐标原点是(500，100)，半径是 50，起始角是 Math.PI/6，结束角是 Math.PI*1.5。绘制代码如下：

```
context.beginPath();
context.arc(500,100,50,Math.PI/6,Math.PI*1.5);
context.closePath();
context.fill();
```

步骤 05 参考上个步骤中的代码，继续通过 arc()绘图，绘图时指定的坐标原点是(100，250)，半径、起始角和结束角不变，但是需要指定逆时针方向绘图。

步骤 06 分别通过 stroke()方法和 fill()方法绘制空心圆形和实心圆形，空心圆形的笔触颜色和实心圆形的填充颜色一致，均为 rgba(0,255,0,0.25)。以空心圆形为例，其代码如下：

```
context.beginPath();
context.strokeStyle = 'rgba(0,255,0,0.25)';
context.arc(300,250,50,0,Math.PI * 2);
context.closePath();
context.stroke();
```

步骤 07 在浏览器中运行上述代码，查看效果，如图 8-9 所示。

此外，可以使用 arcTo()方法在画布上创建介于两个切线之间的弧/曲线，使用 stroke()方法在画布上绘制确切的弧。

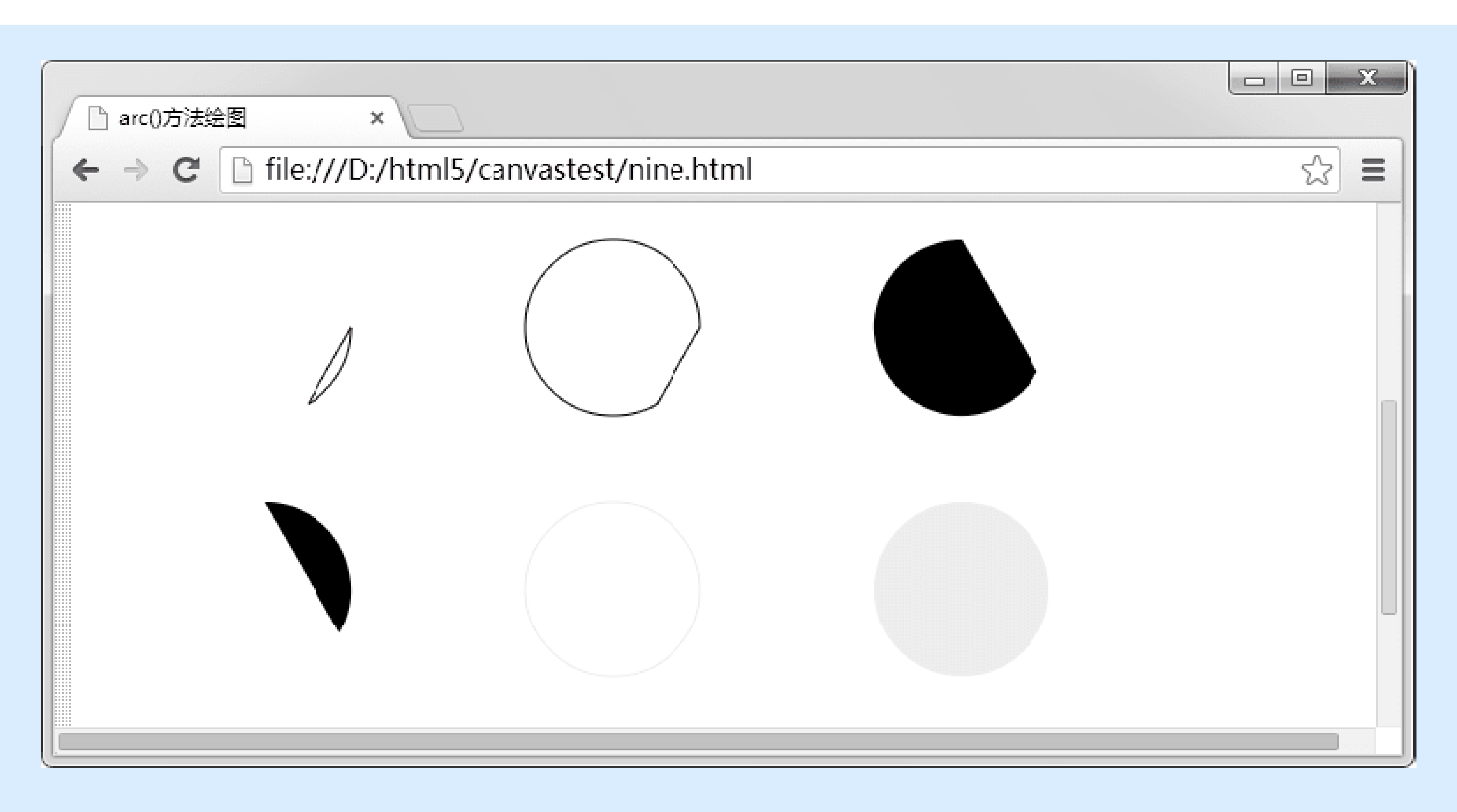


图 8-9　arc()方法的使用

基本语法如下：

```
context.arcTo(x1,y1,x2,y2,r);
```

上述语法中，arcTo()方法的 x1 和 y1 参数表示弧的起点的 x 坐标和 y 坐标；x2 和 y2 参数表示弧的终点的 x 坐标和 y 坐标；r 表示弧的半径。

【例 8.10】

下面的代码演示了 arcTo()方法的基本使用：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.beginPath();
        context.moveTo(20,20);                    // 创建开始点
        context.lineTo(100,20);                   // 创建水平线
        context.arcTo(150,20,150,70,50);          // 创建弧
        context.lineTo(150,120);                  // 创建垂直线
        context.stroke();                         // 进行绘制
    }
}
```

8.4.4 贝塞尔曲线

计算机图形学中，曲线和曲面是重要的组成部分，其中贝塞尔曲线是目前应用广泛的曲线之一。上下文对象提供了两种绘制贝塞尔曲线的方法，它们分别是 quadraticCurveTo()方法和 bezierCurveTo()方法。

1．quadraticCurveTo()方法

quadraticCurveTo()方法通过使用表示二次方贝塞尔曲线的指定控制点，向当前路径添加一个点。二次方贝塞尔曲线需要两个点，第一个点是用于二次贝塞尔计算中的控制点，第二个点是曲线的结束点。曲线的开始点是当前路径中的最后一个点，如果路径不存在，

那么需要使用 beginPath()方法和 moveTo()方法来定义开始点。quadraticCurveTo()方法的基本语法如下：

```
context.quadraticCurveTo(cpx,cpy,x,y);
```

在上述方法中，quadraticCurveTo()方法需要传入 4 个参数：cpx 和 cpy 表示贝塞尔控制点的 x 坐标和 y 坐标；x 和 y 则表示结束点的 x 坐标和 y 坐标。

【例 8.11】

下面的代码演示了 quadraticCurveTo()方法的使用：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.beginPath();
        context.moveTo(20,20);
        context.quadraticCurveTo(20,100,200,20);
        context.stroke();
    }
}
```

在上述代码中，moveTo(20,20)指定开始点，quadraticCurveTo(20,100,200,20)中(20,100)指定控制点；(200,20)指定结束点。

2．bezierCurveTo()方法

bezierCurveTo()方法通过使用表示三次贝塞尔曲线的指定控制点，向当前路径添加一个点。三次贝塞尔曲线需要三个点：前两个点是用于三次贝塞尔计算中的控制点，第三个点是曲线的结束点。曲线的开始点是当前路径中的最后一个点，如果路径不存在，那么可使用 beginPath()和 moveTo()方法来定义开始点。bezierCurveTo()方法的基本语法如下：

```
context.bezierCurveTo(cp1x,cp1y,cp2x,cp2y,x,y);
```

在上述语法中，bezierCurveTo()方法需要传入 6 个参数：cp1x 和 cp1y 分别表示第一个贝塞尔控制点的 x 坐标和 y 坐标；cp2x 和 cp2y 分别表示第二个贝塞尔控制点的 x 坐标和 y 坐标；x 和 y 分别表示结束点的 x 坐标和 y 坐标。

【例 8.12】

下面的代码演示了 bezierCurveTo()方法的使用：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.beginPath();
        context.moveTo(20,20);
        context.bezierCurveTo(20,100,200,100,200,20);
        context.stroke();s  }
}
```

在上述代码中，moveTo(20,20)表示开始点；bezierCurveTo(20,100,200,100,200,20)中的(20,100)指定第一个控制点；(200,100)指定第二个控制点；(200,20)指定结束点。

8.5 图形变换和组合

有时候，用户要求开发者将绘制的图形放大或者缩小，或者进行旋转，这时再使用上面的方法就不能满足这些要求。上下文对象提供了与这些操作有关的方法，它们分别是scale()方法、rotate()方法、translate()方法、transform()方法、setTransform()方法。

8.5.1 图形变形

图形变形涉及到图形的平移、旋转和缩放操作，因此，这涉及到以下 3 个方法。

1．图形平移

图形平移时需要使用到 translate()方法，该方法表示重新映射画布上的(0,0)位置，(0,0)即坐标原点。该方法的基本语法如下：

```
context.translate(x,y);
```

在上述语法中，使用 translate()时需要传入两个参数，第一个参数表示添加到水平坐标 x 上的值，即坐标原点向 x 轴方向平移 x，第二个参数表示添加到垂直坐标 y 上的值，即坐标原点向 y 轴方向平移 y。

【例 8.13】

下面首先通过 fillRect()方法在(10,10)坐标处绘制宽度为 100 像素、高度为 50 像素的矩形，然后平移原点坐标到(70,70)，平移完毕后再次绘制该图。代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.fillRect(10,10,100,50);
        context.translate(70,70);
        context.fillRect(10,10,100,50);
    }
}
```

运行上述代码，查看平移后的效果。需要注意的是，再次绘制新的矩形时，其位置是从(80，80)开始绘制的。

2．图形旋转

rotate()方法旋转当前的图形。基本语法如下：

```
context.rotate(angle);
```

从上述语法中可以看出，使用 rotate()方法时需要传入一个表示旋转角度的参数，其单位是弧度。如果需要将角度转换为弧度，可以使用 degrees*Math.PI/180 公式进行计算。例如，如果需要旋转 5 度，指定的公式是 5*Math.PI/180。默认情况下，旋转的方向为顺时针

方向。

【例 8.14】

下面的代码演示了 rotate()方法的使用：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.rotate(30*Math.PI/180);
        context.fillRect(300,20,100,50);
    }
}
```

3．图形缩放

图形缩放是指图形的缩小或放大效果，实现该功能时需要调用 scale()方法。如果对图形进行缩放，所有之后的绘图也将会被缩放，定位也会被缩放。例如，对于 scale(2,2)来说，绘图时将定位于距离画布左上角两倍远的位置。scale()方法的基本语法如下：

```
context.scale(scalewidth,scaleheight);
```

从上述语法中可以看出，scale()方法使用时需要传入两个参数，第一个参数表示缩放当前绘图的宽度(1=100%，0.5=50%，2=200%，依次类推)，第二个参数表示缩放当前绘图的高度(1=100%，0.5=50%，2=200%，依次类推)。

【例 8.15】

绘制矩形，将其放大到 200%，然后再次绘制矩形。代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.strokeRect(5,5,25,15);
        context.scale(2,2);
        context.strokeRect(5,5,25,15);
    }
}
```

4．平移、旋转和缩放

一般情况下，开发者不会单独地使用一个变形特效，通常会将两个或三个变形结合起来使用，例如同时使用平移和旋转特效。使用多种特效时，使用顺序不同可能导致画出的结果也会有所不同，它们的顺序可能是平移、旋转、缩放；平移、缩放、旋转；缩放、平移、旋转；缩放、旋转、平移；旋转、平移、缩放；旋转、缩放、平移。

例如，图 8-10 显示了坐标轴变化的两两关系图。其中，蓝色代表平移、红色代表旋转、绿色代表旋转。

【例 8.16】

本示例通过调用平移、旋转和缩放的方法绘制一个不规则的相对比较复杂的图形。实

现步骤如下。

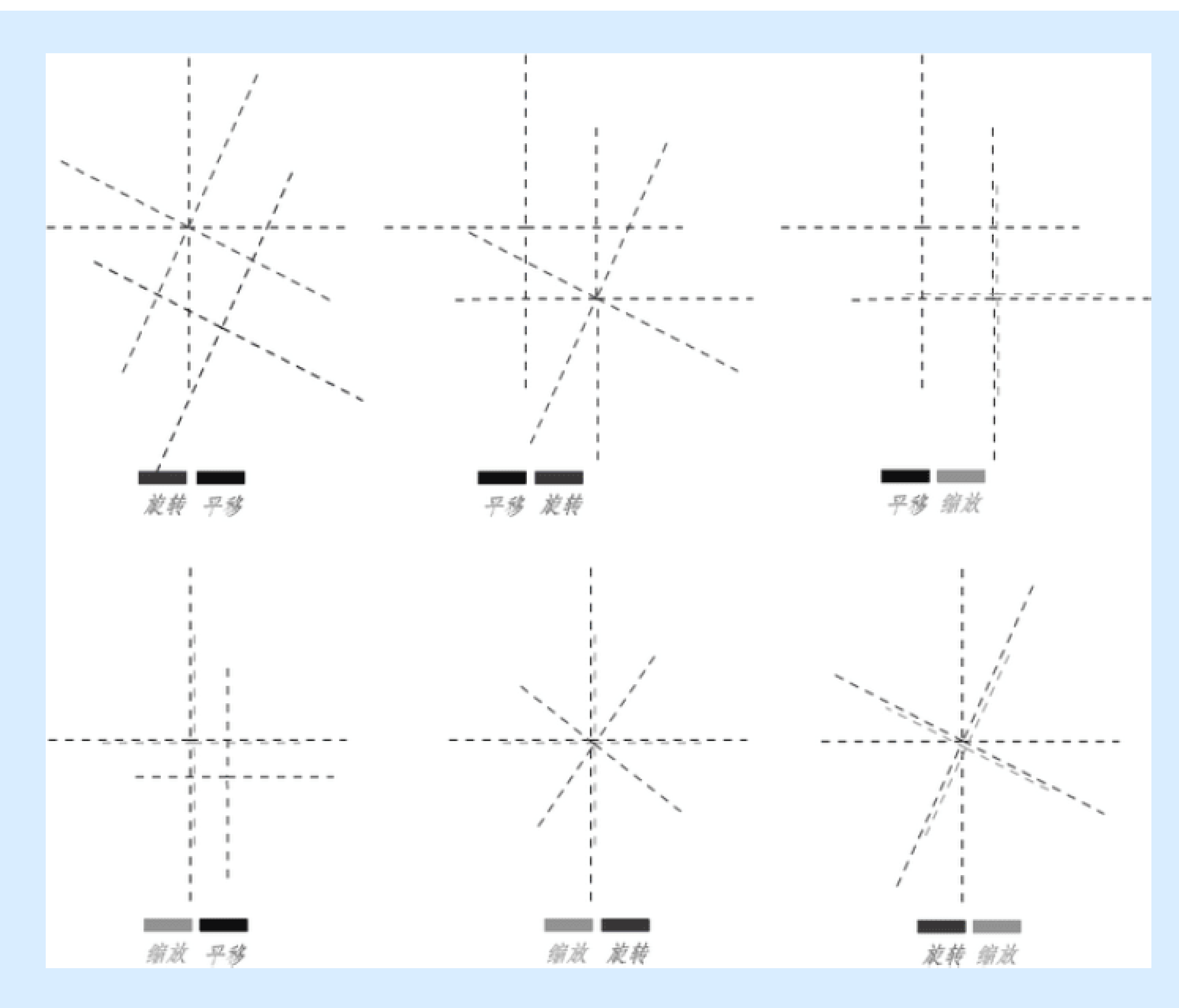


图 8-10　坐标轴两两关系图

步骤 01　向页面中添加宽度为 720 像素、高度为 300 像素的 canvas 元素。

步骤 02　添加 JavaScript 脚本代码，使用三种坐标变换方式绘制图形。代码如下：

```
<script>
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.fillStyle = "white";                       //填充颜色
        context.fillRect(0, 0, 720, 300);                  //绘制矩形
        context.translate(380, 5);                     //将图形平移
        context.fillStyle = "#6CA6CD";                 //填充颜色
        for(var i=0; i<50; i++) {
            context.scale(0.95, 0.95);                 //缩放
            context.translate(35, 25);                 //平移
            context.rotate(Math.PI / 11);                  //旋转
            context.shadowColor = "red";                   //阴影颜色
            context.shadowBlur = 10;                   //阴影模糊路径
            context.fillRect(0, 0, 100, 50);               //绘制矩形
        }
    }
}
</script>
```

上述代码首先获取页面中的 canvas 元素，接着创建上下文对象，通过 fillStyle 属性和 fillRect()绘制一个宽度为 720 像素、高度为 300 像素的矩形，绘制完毕后将其进行平移。然后通过 for 语句进行循环，以此进行缩放、平移和旋转操作，并且设置图形的阴影颜色和模糊路径，最后调用 fillRect()方法循环绘制宽度为 100 像素、高度为 50 像素的矩形。

步骤 03 在浏览器中运行上述代码，查看绘制的图形，如图 8-11 所示。



图 8-11　三种变换方式绘图

技巧：读者可以交换该示例 for 语句中缩放、平移和缩放的代码，将前面列出的 6 种顺序一一进行演示，这时可以发现(平移、旋转、缩放)与(平移、缩放、旋转)效果一样，(缩放、旋转、平移)与(旋转、缩放、平移)效果一样，因此，读者进行实验时只能看到 4 种结果，其实是有两种情况被覆盖了。

8.5.2 矩阵变换

矩阵变换其实是 context 内实现平移、缩放和旋转的一种机制。它的主要原理是矩阵相乘。矩阵变换最常用的一种方法就是 transform()方法，该方法替换该图的当前转换矩阵。基本语法如下：

```
context.transform(a,b,c,d,e,f);
```

从上述语法可以看出，transform()方法使用时需要传入 6 个参数：a 和 b 分别表示水平缩放绘制和水平倾斜绘图；c 和 d 分别表示垂直倾斜绘图和垂直缩放绘图；e 和 f 分别表示水平移动绘图和垂直移动绘图。

使用 transform()方法时，画布上的每个对象都拥有一个当前的变换矩阵，该方法替换当前的变换矩阵。可以使用下面描述的矩阵来操作当前的变换矩阵：

```
a   c   e
b   d   f
0   0   1
```

可使用 context.transform(1,0,0,1,x,y)或 contex.transform(0,1,1,0,x,y)代替 translate(x,y)方法实现平移。在 transform()方法实现平移时，前 4 个参数表示不对图形进行操作，x 和 y 的设置分别表示将坐标原点向右移动 e 个单位，并向下移动 f 个单位。

可以使用 context.transform(x,0,0,y,0,0)或 context.transform(0,y,x,0,0)方法代替 scale(x,y)

方法。在 transform()方法实现缩放时，前面 4 个参数表示将图形横向扩大或缩小 x 倍，纵向扩大或缩小 y 倍，最后两个参数表示坐标原点不移动。

使用 transform()方法实现旋转时要比实现平移和缩放复杂，它可以通过两种设置方式进行实现。

(1) 第一种方式实现旋转，代码如下：

```
context.transform(
            Math.cos(angle*Math.PI/180),
            Math.sin(angle*Math.PI/180),
            -Math.sin(angle*Math.PI/180),
            Math.cos(angle*Math.PI/180),
            0,0);
```

(2) 第二种方式实现旋转，代码如下：

```
context.transform(
            -Math.sin(angle*Math.PI/180),
            Math.cos(angle*Math.PI/180),
            Math.cos(angle*Math.PI/180),
            Math.sin(angle*Math.PI/180),
            0,0);
```

在上述两种方式的代码中，前 4 个参数利用三角函数完整旋转，angle 参数表示按照顺时针旋转的角度，最后两个参数指定为 0，表示坐标原点不发生改变。

【例 8.17】

首先绘制一个宽度为 250 像素、高度为 100 像素的矩形，其填充颜色为黄色。接着通过 transform()方法添加一个新的变换矩阵，再次绘制矩形，然后再次添加一个新的变换矩阵后绘制矩形。这时，每次调用 transform()时，它都会在前一个变换矩阵上构建新图形。实现的 JavaScript 代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");          //创建上下文对象
        context.fillStyle = "yellow";                   //填充颜色为黄色
        context.fillRect(0,0,250,100)                   //绘制矩形
        context.transform(1,0.5,-0.5,1,30,10);          //变换矩阵
        context.fillStyle = "red";                      //填充颜色为红色
        context.fillRect(0,0,250,100);                  //绘制矩形
        context.transform(1,0.5,-0.5,1,30,10);          //变换矩阵
        context.fillStyle = "blue";                     //填充颜色为蓝色
        context.fillRect(0,0,250,100);                  //绘制矩形
    }
}
```

使用 transform()方法后，接着要绘制的图形都会按照移动后的坐标原点与新的变换矩阵相结合的方法进行重置，必要时可以使用 setTransform()方法对变换矩阵进行重置。

setTransform()的基本语法如下：

```
context.setTransform(a,b,c,d,e,f);
```

使用 setTransform()方法时，需要传入 6 个参数，这些参数的含义与 transform()方法一致。简单地说，setTransform()方法允许开发者缩放、旋转、平移并倾斜当前的环境，该变换只会影响 setTransform()方法调用之后的绘图。

【例 8.18】

绘制一个矩形后通过 setTransform()方法重置并创建新的变换矩阵，再次绘制矩形，重置并创建新的变换矩阵，然后再次绘制矩形。JavaScript 实现代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.fillStyle = "yellow";                  //填充颜色为黄色
        context.fillRect(0,0,250,100)                  //绘制矩形
        context.setTransform(1,0.5,-0.5,1,30,10);      //重置变换矩阵
        context.fillStyle = "red";                     //填充颜色为红色
        context.fillRect(0,0,250,100);                 //绘制矩形
        context.setTransform(1,0.5,-0.5,1,30,10);      //重置变换矩阵
        context.fillStyle = "blue";                    //填充颜色为蓝色
        context.fillRect(0,0,250,100);                 //绘制矩形
    }
}
```

上述代码在每次调用 setTransform()时，它都会重置前一个变换矩阵，然后再构建新的矩阵，因此在本示例中不会显示红色矩形，因为它在蓝色矩形下面。

8.5.3 图形组合

在前面的示例中，使用上下文对象可以将一个图形重叠绘制在另一个图形上面，但是图形中能够被看到的部分完全取决于以哪种方式进行组合，这时需要使用图形组合技术。图形组合时涉及到两个属性：globalAlpha 和 globalCompositeOperation。

1．globalAlpha 属性

globalAlpha 属性设置或者返回绘图的当前透明值(alpha 或者 transparency)，该属性值必须是介于 0.0(完全透明)与 1.0(不透明)默认值之间的数。基本语法如下：

```
context.globalAlpha = number;
```

2．globalCompositeOperation 属性

globalCompositeOperation 属性设置或者返回如何将一个源(新的)图形绘制到目标(已有)的图形上。其中，源图形是指 Web 开发者打算放置到画布上的绘图，目标图形是指已经放置在画布上的绘图。基本语法如下：

```
context.globalCompositeOperation = type;
```

为 globalCompositeOperation 属性指定属性值时，该属性值必须是表 8-4 中的一种类型。

3．示例应用

了解 globalAlpha 和 globalCompositeOperation 两个属性之后，下面通过一个示例演示它们的使用及其实现的效果。

表 8-4　globalCompositeOperation 属性的取值

属性取值	说　明
source-over	默认值，在目标图形上显示源图形
source-over	默认设置，表示新图形会覆盖在原有图形之上
destination-over	会在原有图形之上绘制新图形
source-in	新图形会仅仅出现与原有图形相重叠的部分，其他区域都变成透明的
destination-in	原有图形中与新图形重叠的部分会被保留，其他区域都变成透明的
source-out	只有新图形中与原有内容不重叠的部分会被绘制出来
destination-out	原有图形中与新图形不重叠的部分会被保留
source-atop	只绘制新图形中与原有图形重叠的部分。未被重叠覆盖的原有图形，新图形的其他部分变成透明
destination-atop	只绘制原有图形中被新图形重叠覆盖的部分。新图形的其他部分，原有图形中的其他部分变成透明，不绘制新图形中与原有图形相重叠的部分
lighter	两图形中重叠部分做加色处理
darker	两图形中重叠的部分做减色处理
xor	重叠的部分会变成透明
copy	只有新图形会被保留，其他都被清除掉

【例 8.19】

本例根据用户选择的内容，设置图形的组合效果。实现步骤如下。

步骤 01　向 HTML 页面中添加 12 个单选按钮，这些按钮分别表示不同的组合类型。部分代码如下：

```
<input type="radio" name="cktype" value="source-over" checked />source-over
  <input type="radio" name="cktype" value="source-in" />source-in
  <input type="radio" name="cktype" value="source-out" />source-out
```

步骤 02　添加用于提交用户选择信息的普通按钮，并为该按钮添加单击时的事件。代码如下：

```
<input type="button" value="查看效果" onClick="javascript:ChangeChoose()" />
```

步骤 03　创建 ChangeChoose()函数，该函数根据用户选择的组合类型进行设置。实现代码如下：

```
function ChangeChoose(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        var typelist = document.getElementsByName("cktype");
        for(var i=0; i<typelist.length; i++){
            if(typelist[i].checked){
                context.fillStyle="#FF7256";                //填充颜色
                context.fillRect(10,10,60,60);              //绘制矩形
                context.globalAlpha = 0.4;                  //设置透明度
```

```
                    //设置组合类型
                    context.globalCompositeOperation=typelist[i].value;
                    context.beginPath();                           //绘制路径
                    context.fillStyle="#8FBC8F";                   //填充颜色
                    context.arc(60,60,30,0,Math.PI*2,false);       //绘制圆形
                    context.closePath();                           //关闭路径
                    context.fill();                            //将圆形绘制到画布
                }
            }
        }else{
            alert("您的浏览器不支持 canvas");
        }
}
```

上述代码首先获取 canvas 对象和上下文对象，接着获取页面中的所有的单选项，并通过 for 语句进行遍历。在 for 语句中，如果某一项被选中，则分别绘制矩形和圆形，并且指定透明度和组合类型。透明效果为 0.4，组合类型则是当前选中的类型。

步骤 04　在浏览器中运行本例的代码进行测试，lighter 组合效果如图 8-12 所示。

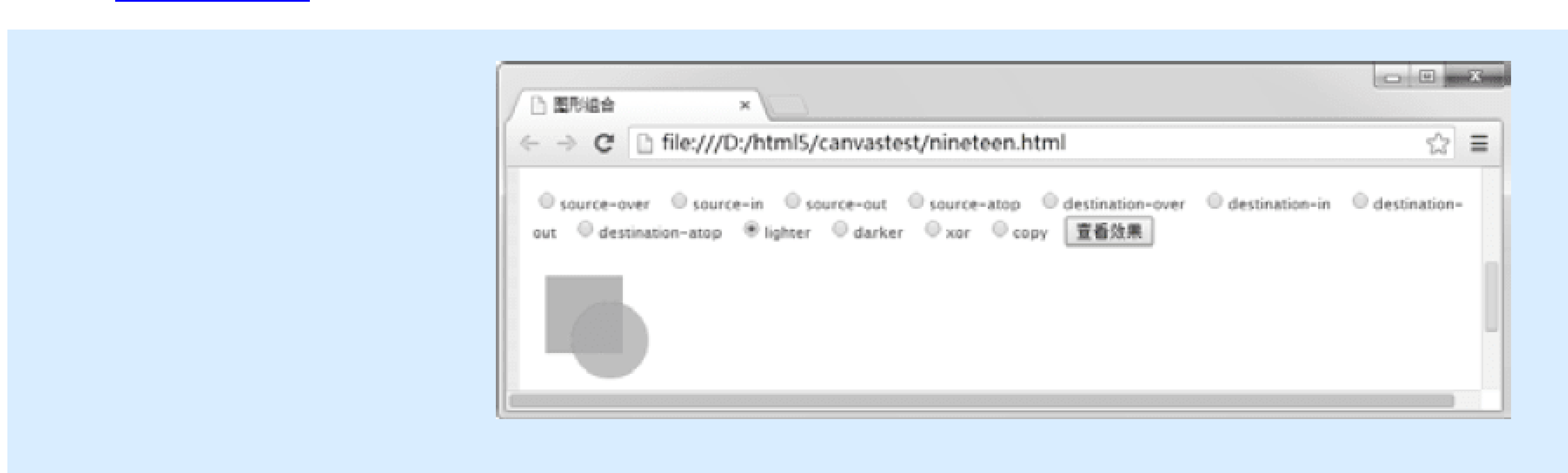


图 8-12　lighter 组合效果

8.6　绘制颜色渐变

渐变是指在填充时从一种颜色慢慢过渡到另一种颜色。一般情况下，可以将渐变分为两种：线性渐变和径向渐变，径向渐变又可以称为放射性渐变。

8.6.1　线性渐变

线性渐变是沿着一根轴线(水平或者垂直)改变颜色，从起点到终点颜色进行顺序渐变(从一边拉向另一边)。上下文对象提供 createLinearGradient()方法创建线性的渐变对象，渐变可用于填充矩形、圆形、线条和文本等。createLinearGradient()的基本语法如下：

```
context.createLinearGradient(x0,y0,x1,y1);
```

从上述语法中可以看出，createLinearGradient()方法使用时需要传入 4 个参数。其中，x0 表示渐变开始点的 x 坐标；y0 表示渐变开始点的 y 坐标；x1 表示渐变结束点的 x 坐标；y1 表示渐变结束点的 y 坐标。

通过 createLinearGradient()方法，只是创建了一个使用两个坐标点的 LinearGradient 对象。如果要设置渐变的颜色，则需要通过 addColorStop()方法。该方法指定渐变对象中的

颜色和位置，它与 createLinearGradient()方法和 createRadialGradient()方法一起使用。

addColorStop()方法的基本语法如下：

```
gradient.addColorStop(stop,color);
```

使用 addColorStop()方法时，需要传入两个参数：stop 参数指定颜色离开渐变起始点的偏移量，它的值位于 0 和 1 之间；color 参数指定结束位置显示的 CSS 颜色值。

例如，图 8-13 通过图形的方式来描述偏移量的含义，0 表示起始点，1 表示结束点。

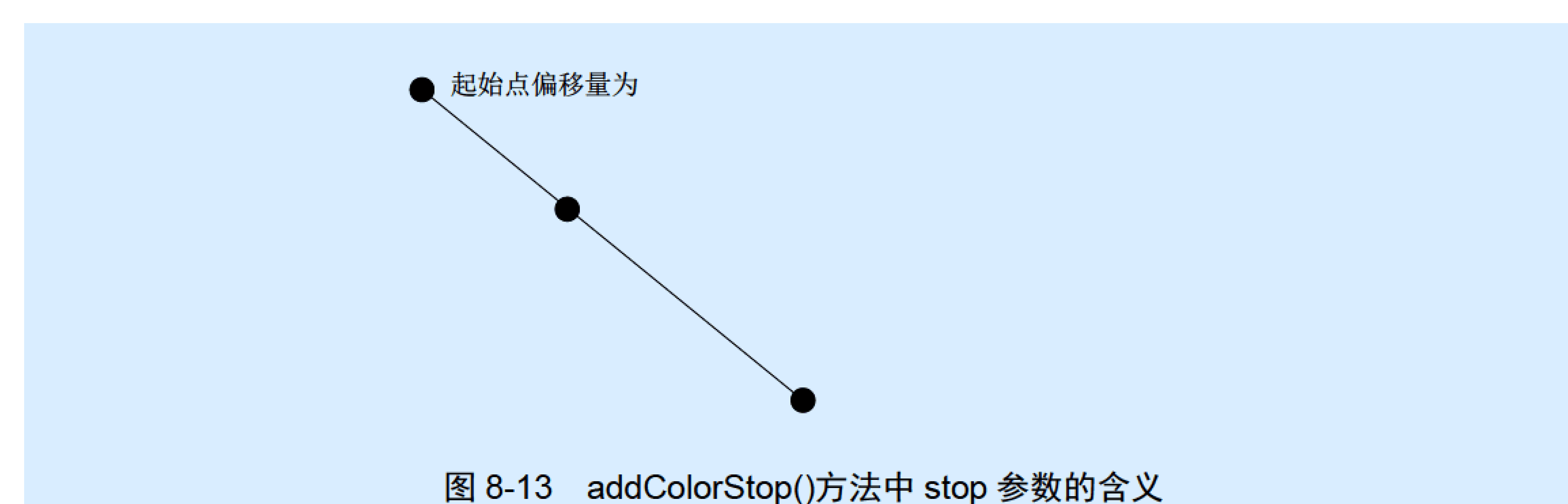


图 8-13　addColorStop()方法中 stop 参数的含义

技巧：使用者可以多次调用 addColorStop()方法来改变渐变。如果不对渐变对象使用该方法，那么渐变将不可见。因此，为了获得可见的渐变，开发者至少需要创建一个色标。

【例 8.20】

本例将 createLinearGradient()方法和 addColorStop()方法结合起来，实现颜色依次从 black 到 magenta、blue、green、yellow 的渐变。实现步骤如下。

步骤 01　向 HTML 页面中添加宽度为 720、高度为 150 的 canvas 元素。

步骤 02　向画布中绘制颜色的线性渐变，实现多种颜色过渡。代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        //创建 LinearGradient 对象
        var gradient = context.createLinearGradient(0,0,500,0);
        gradient.addColorStop(0,"black");
        gradient.addColorStop("0.3","magenta");
        gradient.addColorStop("0.5","blue");
        gradient.addColorStop("0.6","green");
        gradient.addColorStop("0.8","yellow");
        gradient.addColorStop(1,"red");
        context.fillStyle = gradient;              //将填充颜色设置为渐变
        context.fillRect(100,20,500,150);          //绘制矩形
    }
}
```

步骤 03 在浏览器中运行上述代码，查看效果，如图 8-14 为渐变效果。

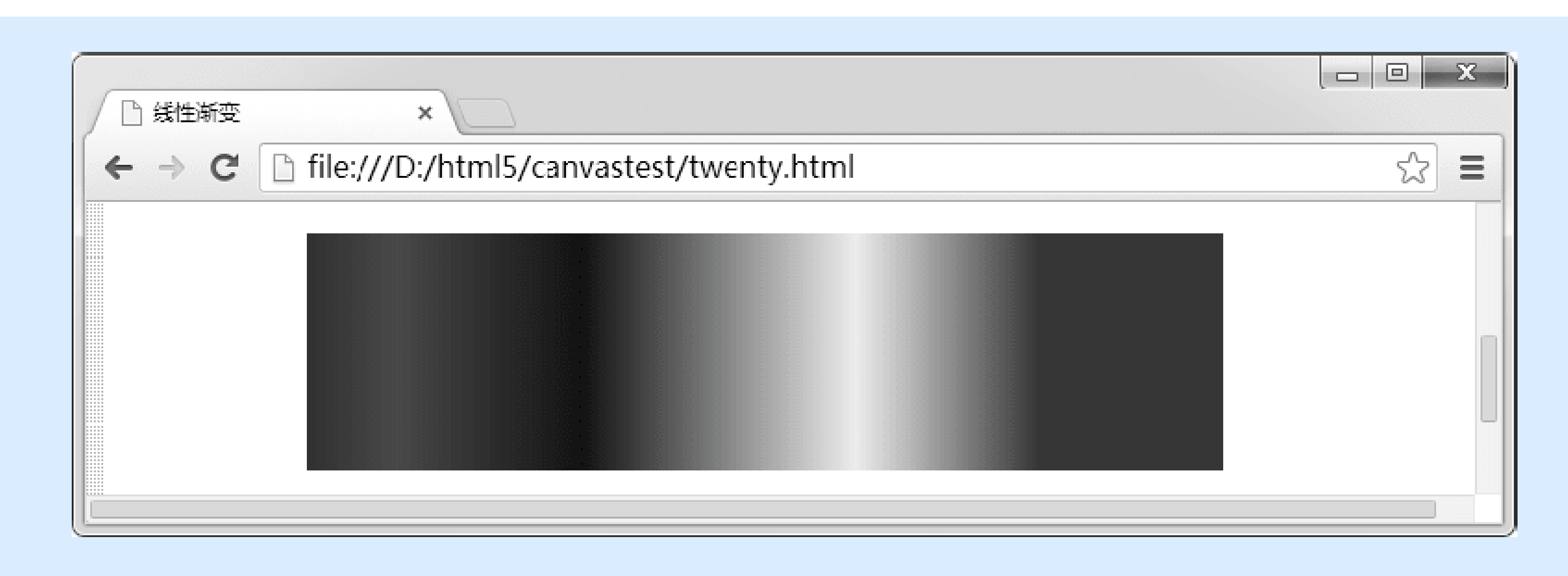


图 8-14 线性渐变的效果

8.6.2 径向渐变

径向渐变是指从起点到终点颜色从内到外进行的圆形渐变(从中间向外拉)。例如，绘制太阳时，沿着太阳的半径方向向外扩散出去的光晕就是一种径向渐变。绘制径向渐变时需要使用 createRadialGradient()方法，基本语法如下：

```
context.createRadialGradient(x0,y0,r0,x1,y1,r1);
```

使用 createRadialGradient()方法绘制径向渐变时，涉及到 6 个参数：x0 和 y0 分别表示渐变开始圆的圆心 x 坐标和 y 坐标；r0 表示渐变开始圆的半径；x1 和 y1 分别表示渐变结束圆的圆心的 x 坐标和 y 坐标；r1 表示结束圆的半径。

径向渐变分别指定了两个圆的大小与位置，从第一个圆的圆心外向外进行扩散渐变，一直扩散到第二个圆的外轮廓处。与线性渐变相同，径向渐变设置颜色时需要使用 addColorStop()方法，同样也需要设置 0~1 之间的浮点数作为渐变转折点的偏移量。

【例 8.21】

在绘制径向渐变时，两个圆的位置可能是同心圆，也可以是不同心的圆。本例绘制两个径向渐变，第一个渐变是同心圆，第二个渐变圆心不同。相关的 JavaScript 代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        var g1 = context.createRadialGradient(
                200, 150, 0, 200, 150, 100);          //圆心相同
        g1.addColorStop(0.1, 'rgb(255,0,0)');
        g1.addColorStop(1, 'rgb(50,0,0)');
        context.fillStyle = g1;
        context.beginPath();
        context.arc(200, 150, 100, 0, Math.PI*2, true);    //绘制圆形
        context.closePath();
        context.fill();
        var g2 = context.createRadialGradient(
                350, 150, 10, 500, 150, 50);          //圆心不同
        g2.addColorStop(0.1, 'rgb(255,0,0)');
        g2.addColorStop(0.5, 'rgb(0,255,0)');
```

```
        g2.addColorStop(1, 'rgb(0,0,255)');
        context.fillStyle = g2;
        context.fillRect(0, 0, 580, 300);              //绘制矩形
    }
}
```

在浏览器中运行上述代码查看效果，如图 8-15 所示。

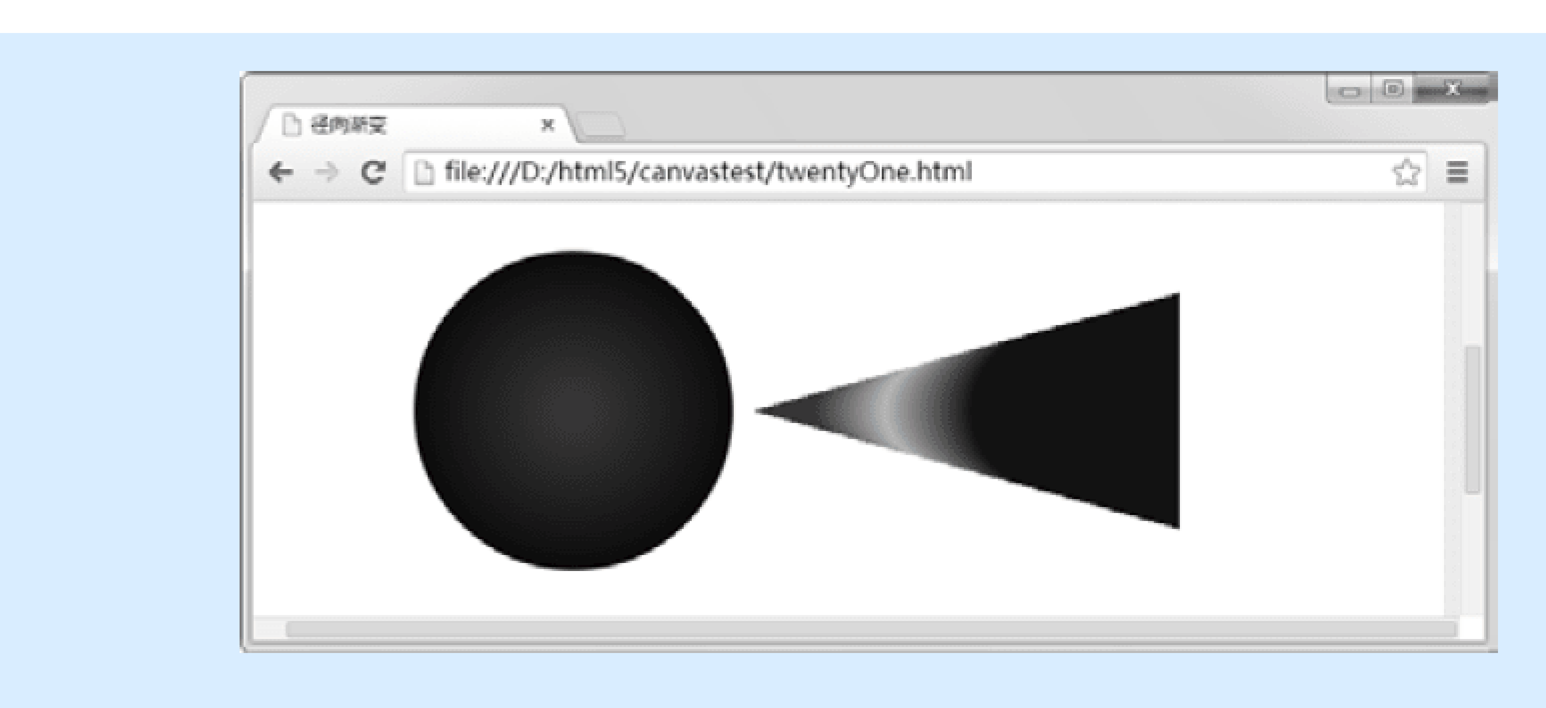


图 8-15　径向渐变的效果

8.7　图片的常用操作

使用上下文对象的方法，不仅可以绘制文本和图形，还可以读取磁盘或者网络中的图片，本节简单了解如何使用上下文对象的方法操作图片。

8.7.1　drawImage()绘制

drawImage()方法表示对图片的绘制，使用该方法可以绘制图片的某一部分，也可以添加或者减少图片的尺寸。通常情况下，drawImage()方法包含 3 种使用语法。

1．drawImage(image,dx,dy)

这是一种最常用的绘制方法，使用时需要传入 3 个参数：第一个参数是指 image 对象，它不仅指向一个 img 元素，还可以是 video 元素或者 JavaScript 中的 image 对象；第二个参数是指目标 x 坐标，即在画布绘制时的横坐标；最后一个参数是指目标 y 坐标，即在画布绘制时的纵坐标。

【例 8.22】

下面通过一个具体的步骤演示 drawImage(image,dx,dy)的使用。步骤如下。

步骤 01　向页面中添加一张图片和一个 canvas 元素。代码如下：

```
<img src="images/ta.jpg" id="img">
<canvas id="MyCanvas" width="500" height="250">当前浏览器不支持 canvas 元素
</canvas>
```

步骤 02　添加 JavaScript 脚本代码，直接调用 drawImage(image,dx,dy)方法绘制图片，指定绘制图片时的起始点的坐标(150,0)。代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        var image = document.getElementById("img");
        context.drawImage(image,150,0);
    }
}
```

步骤 03 在浏览器中运行上述代码查看绘图效果，如图 8-16 所示。

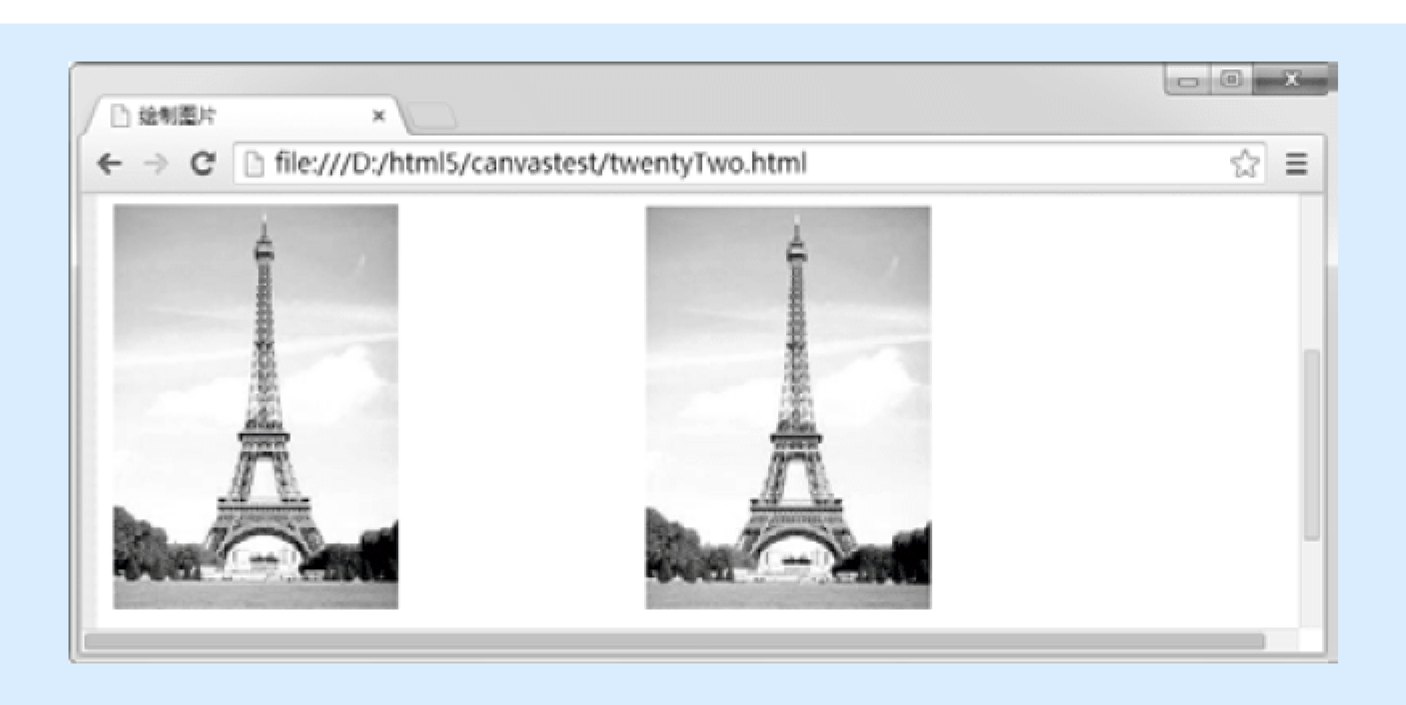


图 8-16　用 drawImage(image,dx,dy)绘制图片

提示：通过 drawImage(image,dx,dy)方法绘制图片时，如果图片的高度小于或等于画布的高度，那么绘制时图片正常显示。如果图片的高度大于画布的高度，即画布高度不够，那么将会绘制图片的一部分。读者可以更改上述示例中的画布高度，更改完毕后再次查看绘制效果。

2．context.drawImage(img,dx,dy,width,height)

与上一种方式相比，这种方式多出了两个参数，width 和 height 分别表示绘制时图像的宽度和高度。例如，更改例 8.22 中的代码，指定绘制图片时的宽度和高度都为 150 像素。代码如下：

```
context.drawImage(image,150,0,150,150);
```

重新刷新浏览器查看绘制效果，如图 8-17 所示。

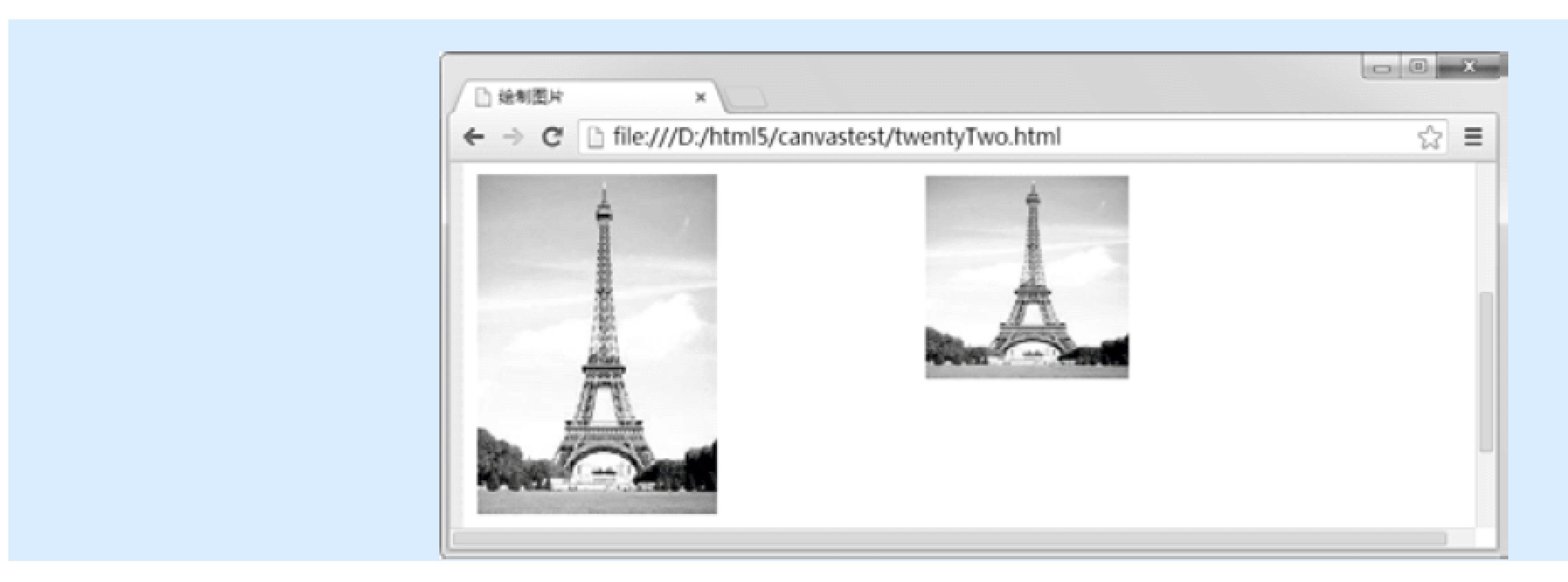


图 8-17　指定图片的宽度和高度

3．context.drawImage(img,sx,sy,swidth,sheight,dx,dy,width,height);

这种语法方式表示剪切图像，并在画布上定位被剪切的部分。在上述方法中指定了 9 个参数。其中，img 指定要使用的图像、画布或者视频；sx 和 sy 是可选参数，指定开始剪切时 x 坐标和 y 坐标的位置；swidth 和 sheight 是可选参数，指定被剪切图片的宽度和高度；dx 和 dy 分别表示在画布上放置图片的 x 坐标和 y 坐标的位置；width 和 height 是可选参数，要使用的图片的宽度和高度(伸展或者缩小图片)。

例如，继续在前面示例的基础上更改内容。代码如下：

```
context.drawImage(image,20,20,150,100,150,0,150,250);
```

重新刷新浏览器查看效果，如图 8-18 所示。

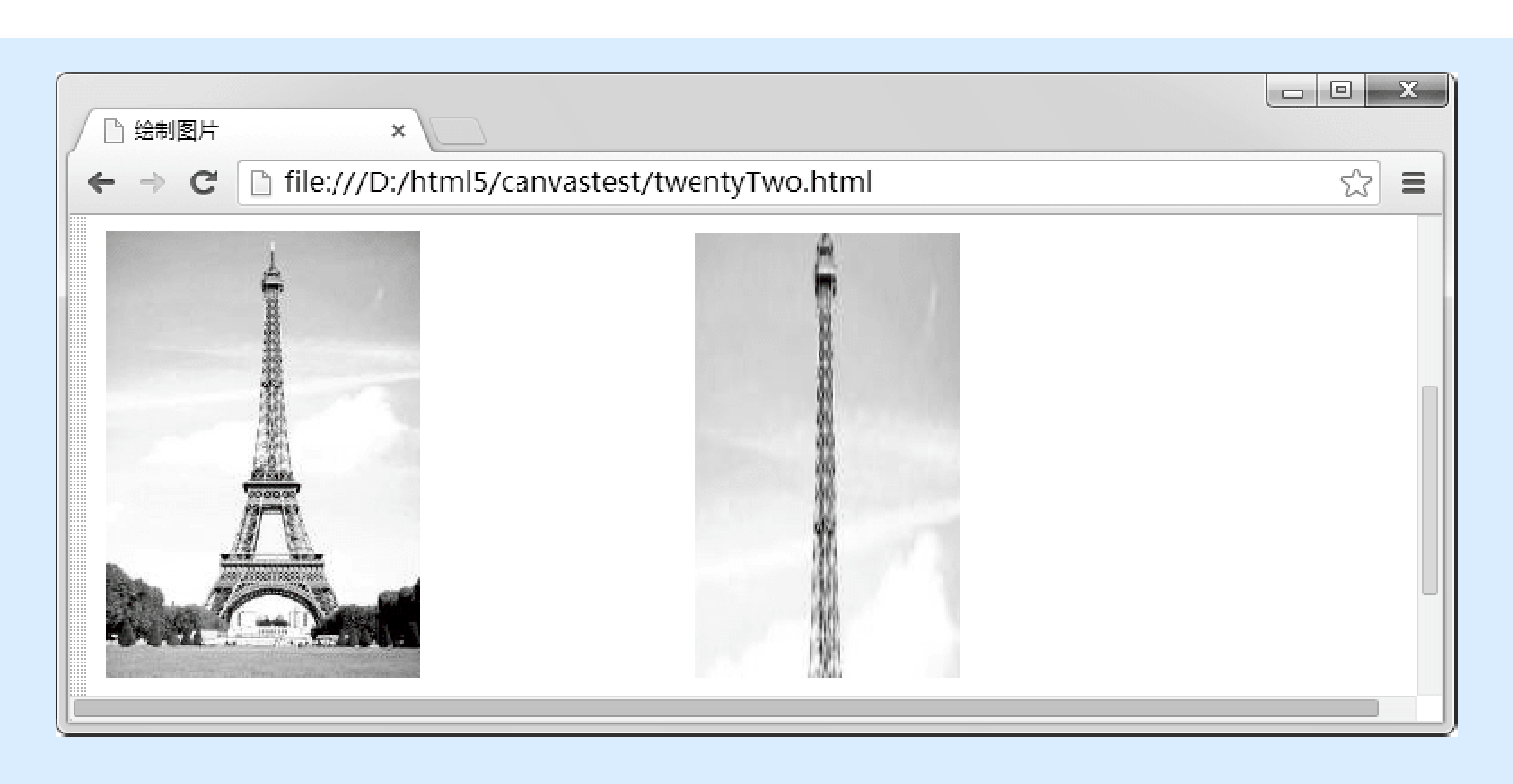


图 8-18　剪切图片的效果

8.7.2　createPattern()方法

createPattern()方法表示在指定的方向上重复指定的元素，该方法通常用于对图片的操作。基本语法如下：

```
context.createPattern(image,"repeat|repeat-x|repeat-y|no-repeat");
```

使用 createPattern()方法时需要传入两个参数：第一个参数指定使用的图片是视频等元素；第二个参数指定平铺方式，它的取值说明如下。

- repeat：默认方式，该方式在水平和垂直方向重复。
- repeat-x：只在水平方向重复。
- repeat-y：只在垂直方向重复
- no-repeat：不重复，该方式只显示一次。

【例 8.23】

根据用户选择的平铺方式查看 createPattern()方法的作用效果。具体的实现步骤如下。

步骤 01　首先添加 4 个单选按钮和一个操作按钮，并为操作按钮添加 Click 事件。相关代码如下：

```
<input type="radio" name="picrepeat" checked value="repeat" />repeat
```

```
 <input type="radio" name="picrepeat" value="repeat-x" />repeat-x
 <input type="radio" name="picrepeat" value="repeat-y" />repeat-y
 <input type="radio" name="picrepeat" value="no-repeat" />no-repeat
 <input type="button" value="设置效果" onClick = "SetRepeat()"/>
```

步骤 02　创建一个隐藏的图片元素和一个 canvas 元素，并指定 canvas 元素的宽度和高度。代码如下：

```
<img src="images/ta.jpg" id="img" height="100" width="100" hidden>
<canvas id="MyCanvas" width="720" height="500">当前浏览器不支持 canvas 元素
</canvas>
```

步骤 03　创建 SetRepeat()函数，该函数获取用户选择的平铺样式，并且通过 createPattern()方法指定平铺。代码如下：

```
function SetRepeat(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        var image = document.getElementById("img");     //获取页面中的图片
        var checks = document.getElementsByName("picrepeat");  //获取选择项
        for(var i=0; i<checks.length; i++){                   //遍历选择项
            if(checks[i].checked) {                         //如果当前项被选中
                context.clearRect(0,0,720,400);                 //清空矩形
                //设置平铺效果
                var cx = context.createPattern(image,checks[i].value);
                context.fillStyle = cx;                         //填充样式
                context.fillRect(0,0,720,400);                  //绘制矩形
            }
        }
    }
}
```

步骤 04　在浏览器中运行上述代码，选择不同的平铺样式查看效果，如图 8-19 所示为默认的平铺效果。

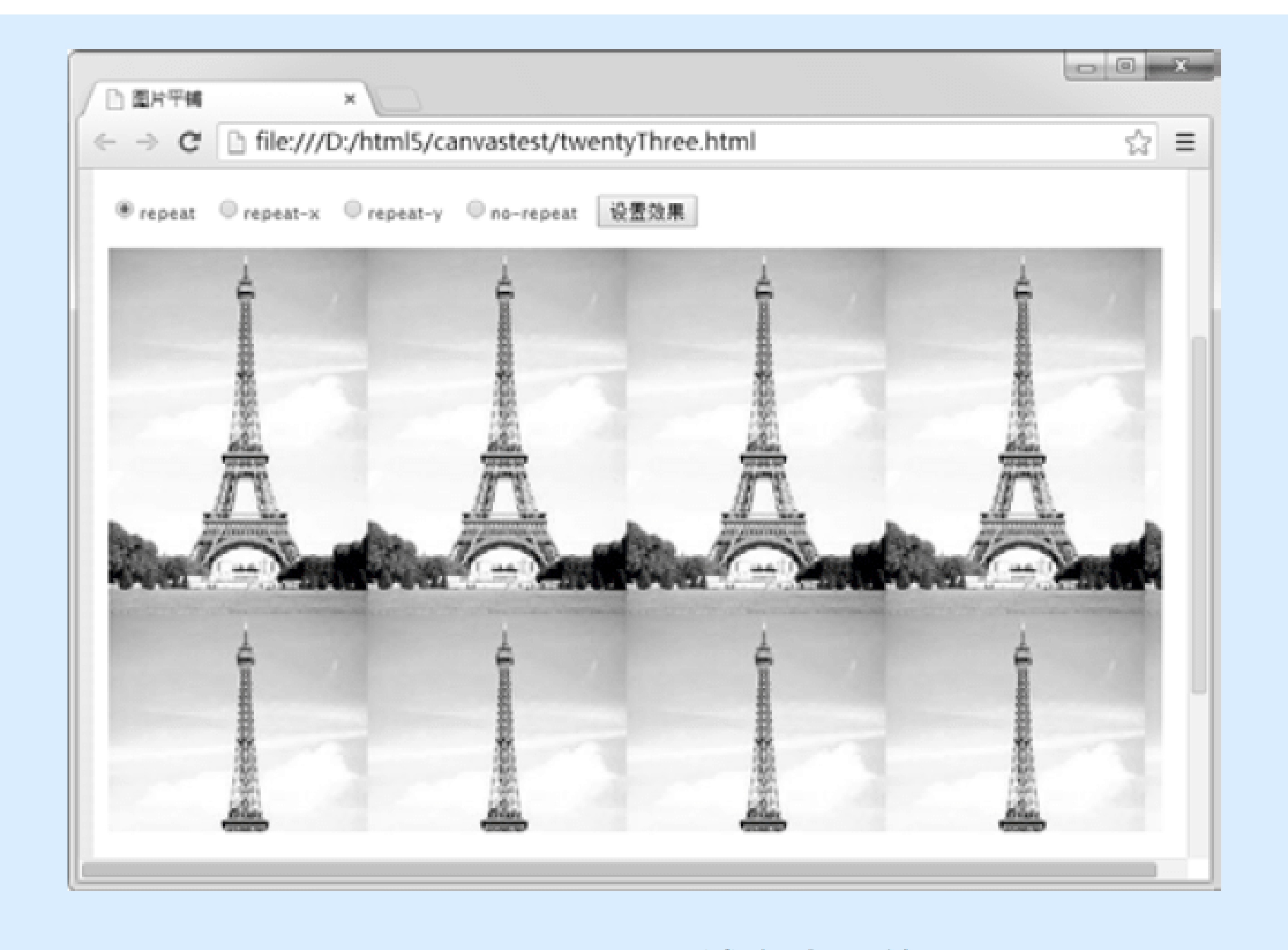


图 8-19　repeat 平铺方式的效果

8.7.3 clip()方法

clip()方法表示从原始画布中剪切任意形状和尺寸。一旦剪切了某个区域，则所有之后的绘图都会被限制在被剪切的区域内(不能访问画布上的其他区域)。当然，Web 开发者可以在使用 clip()方法前通过 save()方法对当前画布区域进行保存，并在以后的任意时间对其进行恢复(通过 restore()方法)。

clip()方法的使用非常简单，直接调用即可。基本语法如下：

```
context.clip();
```

【例 8.24】

本例通过 clip()方法裁剪一张图片，裁剪后的图片是一个星形。实现步骤如下。

步骤 01 向页面中分别添加 img 元素和 canvas 元素，页面代码不再给出。

步骤 02 添加 JavaScript 代码，在这段代码中首先绘制宽度为 350 像素、高度为 300 像素的矩形，接着创建图片元素，并且在图片的 load 事件中调用自定义的 drawImage()函数。代码如下：

```
window.onload = function(){
    var canvas = document.getElementById("MyCanvas");
    if(canvas.getContext){
        var context = canvas.getContext("2d");
        context.fillStyle = "white";
        context.fillRect(0, 0, 350, 300);                //矩形
        image = new Image();
        image.onload = function () {
            drawImg(context,image);
        }
        image.src = "images/flower.jpg";                 //图片路径
    }
}
```

步骤 03 drawImg()函数需要传入两个参数：第一个参数是上下文对象，第二个参数是图片对象。在该方法中调用 create5StarClip()裁剪星形图片，并且调用 drawImage()方法绘图。代码如下：

```
function drawImg(context, image) {
   create5StarClip(context);
   context.drawImage(image,0,0);                  //绘制图片
}
```

步骤 04 create5StarClip()函数将图片裁剪为星形，这里用到数学知识。代码如下：

```
function create5StarClip(context) {
   var n = 0,dx = 180,dy = 135,s = 150;
   context.beginPath();                           //开始绘制路径
   var x = Math.sin(0);
   var y = Math.cos(0);
   var dig = Math.PI / 5 * 4;
   for (var i=0; i<5; i++) {                      //遍历绘制五星的 5 个点
      var x = Math.sin(i * dig);
```

```
        var y = Math.cos(i * dig);
        context.lineTo(dx + x * s, dy + y * s); //绘制线条
    }

    context.closePath();                    //关闭路径
    context.clip();                         //裁剪
}
```

步骤 05 在浏览器中运行示例代码，查看裁剪效果，如图 8-20 所示。

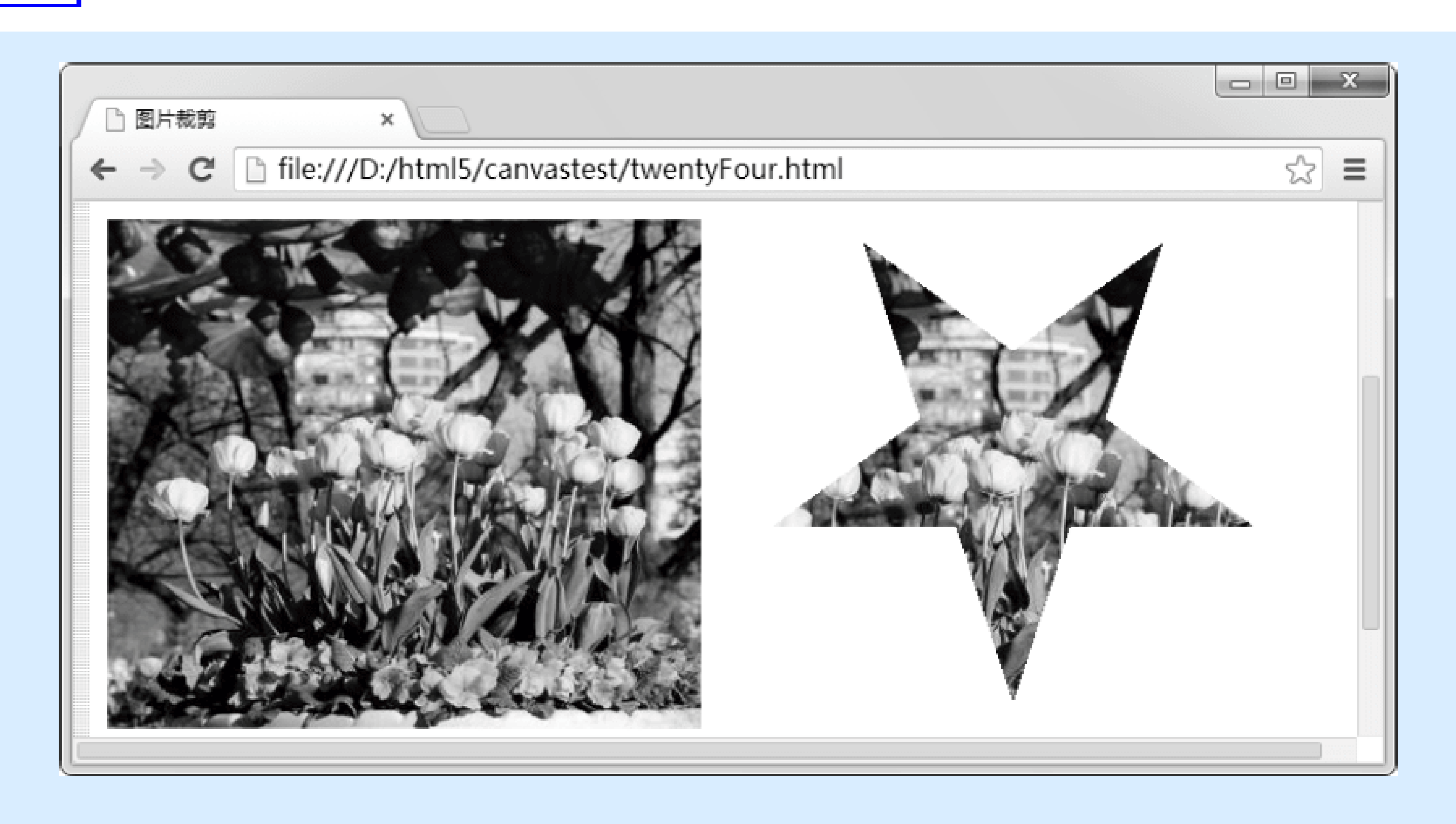


图 8-20　图片裁剪效果

8.8　实现动画特效

前面介绍的内容都是静态的，即没有实现任何特效。例如，Web 开发者可以通过调用上下文对象的方法绘制一个静态钟表，当然，也可以借助于有关的 JavaScript 知识实现动画特效，让钟表“动”起来。

8.8.1　了解动画

由于是用 JavaScript 脚本操作 canvas 的，因此，要实现一些交互动画是很容易的。只不过 canvas 并非是为了制作动画而出现的，因此没有动画制作中帧的概念。使用定时器不断地绘制 canvas 画面成为实现动画效果的通用解决方式。

JavaScript 中的 setInterval(code, millisec)方法可以按照指定的时间间隔 millisec 来反复调用 code 所指向的函数或者代码串。

基本语法如下：

```
setInterval(code, millisec);
```

上述语法中 code 参数表示执行动画的函数，millisec 参数为时间间隔，单位是毫秒。这样，通过将绘图函数作为第一个参数传给 setInterval()方法，在每次被调用的过程中移动画面中图形的位置，来最终实现一种动画的体验。

注意：虽然 setInterval()方法的第二个参数允许开发者对绘图函数的调用频率进行设定，但这始终都是一种最为理想的情况。由于这种绘图频率很大程度上取决于支持 canvas 的底层 JavaScript 引擎的渲染速度以及相应绘图函数的复杂性，因而实际运行的结果往往都是要慢于指定绘图频率的。

8.8.2 实战——绘制动态闪动线条

JavaScript 的功能非常强大，本节实战利用 canvas 元素、上下文对象以及 setInterval()方法绘制动态线条，线条的颜色随机设置。实现步骤如下。

步骤 01 向页面中添加 canvas 元素，并且指定其宽度、高度和样式。代码如下：

```
<canvas id="MyCanvas" width="400" height="400" style="border: 10px
#FFDAB9 solid">当前浏览器不支持 canvas 元素</canvas>
```

步骤 02 创建 JavaScript 脚本，首先判断当前浏览器是否支持 canvas 元素，如果支持则绘制彩色线条。代码如下：

```
var mycanvas  = document.getElementById("MyCanvas");
if(mycanvas.getContext){
    var mycontext = mycanvas.getContext("2d");
    var x,y,x2,y2,r,g,b;                //定义变量
    function line() {
        x=Math.floor(Math.random()*190) + Math.floor(Math.random()*190);
        y=Math.floor(Math.random()*190) + Math.floor(Math.random()*190);
        x2=Math.floor(Math.random()*190)+ Math.floor(Math.random()*190);
        y2=Math.floor(Math.random()*190)+ Math.floor(Math.random()*190);
        r=Math.floor(Math.random()*255);
        g=Math.floor(Math.random()*255);
        b=Math.floor(Math.random()*255);
        mycontext.moveTo(x, y);         //原点
        mycontext.lineTo(x2, y2);       //目标点
        mycontext.strokeStyle = 'rgb(' + r + ',' + g +  ',' + b + ')';
        mycontext.lineWidth = Math.floor(Math.random()*6);
        mycontext.stroke();
        mycontext.restore();
    }
}
```

上述代码定义 7 个变量，这些变量在 line()函数中进行调用。x 和 y 表示开始绘制线条时起始位置的横坐标和纵坐标；x2 和 y2 表示线条结束点的横坐标和纵坐标；r、g、b 则定义 R(Red)、G(Green)、B(Blue)颜色。在 line()函数中首先设置 x、y 以及 x2、y2 等变量的值，然后再调用 moveTo()和 lineTo()方法来绘制线条。

步骤 03 通过 setInterval()函数设置计时特效，每 100 毫秒调用一次 line()函数。代码如下：

```
setInterval(line, 100);
```

步骤 04 在浏览器中运行代码，查看效果，如图 8-21 所示为动态闪动线条的一个瞬

时特效。

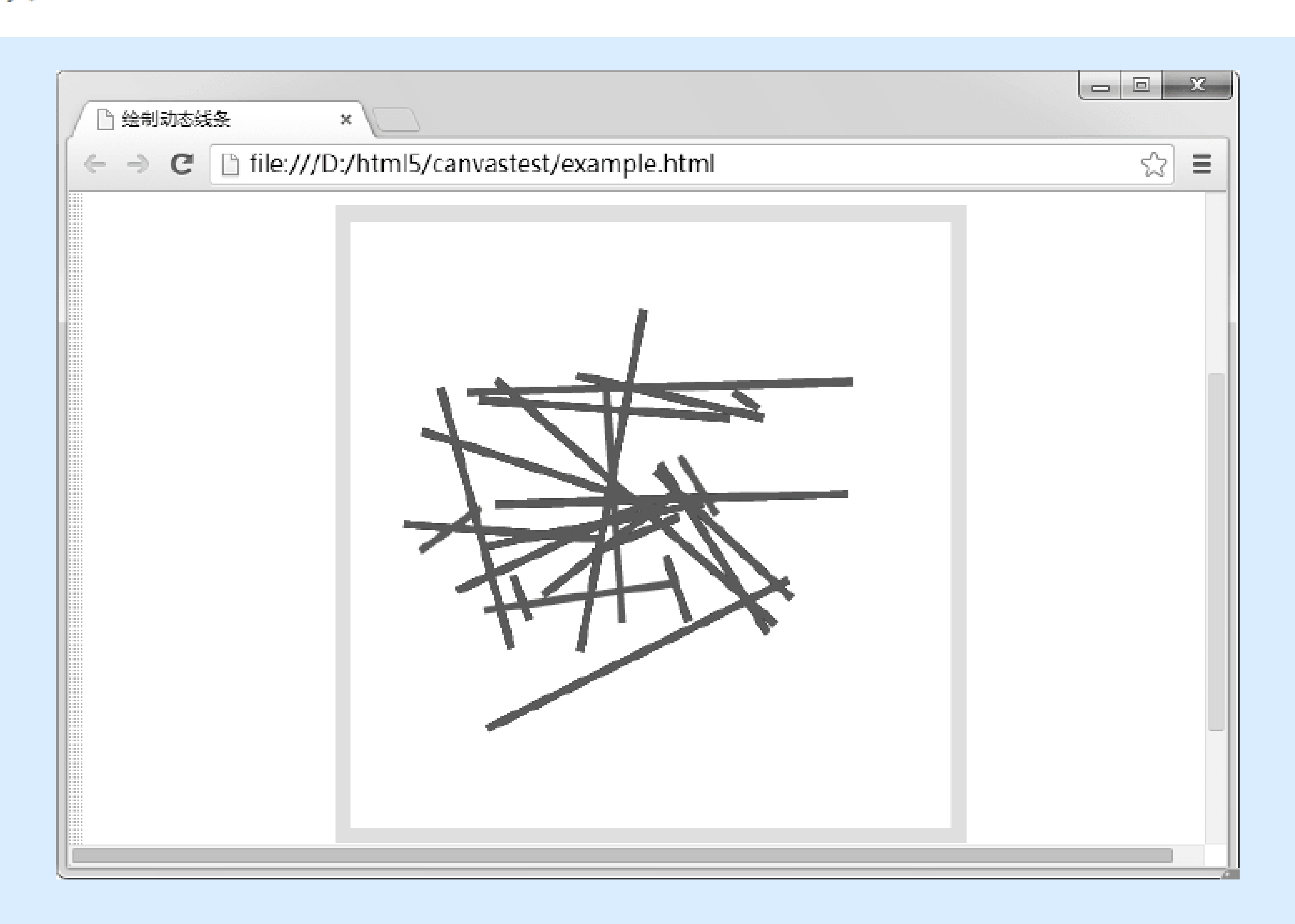


图 8-21　动态闪动线条的瞬时特效

8.9　本 章 习 题

1. 填空题

(1)　获取上下文对象时需要通过 canvas 对象的__________方法。

(2)　绘制文本时，__________方法表示在画布上绘制“被填充”的文本。

(3)　调用上下文对象的__________方法用于清空指定的矩形区域。

(4)　__________方法表示起始一条路径，或者重置当前路径。

(5)　绘制贝塞尔曲线涉及到的两个方法是__________和 bezierCurveTo()。

2. 选择题

(1)　上下文对象可以调用__________属性设置阴影的模糊程度。

A．shadowColor

B．shadowBlur

C．shadowOffsetX

D．shadowOffsetY

(2)　绘制图形完毕后，可以调用__________方法关闭路径。

A．startPath()

B．clip()

C．beginPath()

D．closePath()

(3) 绘制线性渐变和径向渐变时都需要调用__________方法追加颜色渐变效果。

A．createColorStop()

B．createLinearGradient()

C．addColorStop()

D．createRadialGradient()

(4) createPattern()方法实现图片平铺时，默认的平铺方式是__________。

A．repeat

B．repeat-x

C．repeat-y

D．no-repeat

3. 上机练习

(1) 绘制指定的图形

根据本章介绍的上下文对象的方法创建不同的图形，包括线条、三角形、矩形、圆形和扇形。除此之外，还需要使用任意一种绘制文本的方法绘制出文本字符串，该字符串是"我在练习使用绘制文本"。

(2) 绘制指定的图形

利用本章介绍的知识绘制出一个比较复杂的图形，该图形的最终效果如图 8-22 所示。注意：在绘制图形时可能会使用到数学知识。

部分代码如下：

```
var x = Math.sin(0);
var y = Math.cos(0);
var dig = Math.PI / 15 * 11;
for (var i=0; i<30; i++) {
    var x = Math.sin(i * dig);
    var y = Math.cos(i * dig);
    context.lineTo(dx + x * s, dy + y * s);
}
```

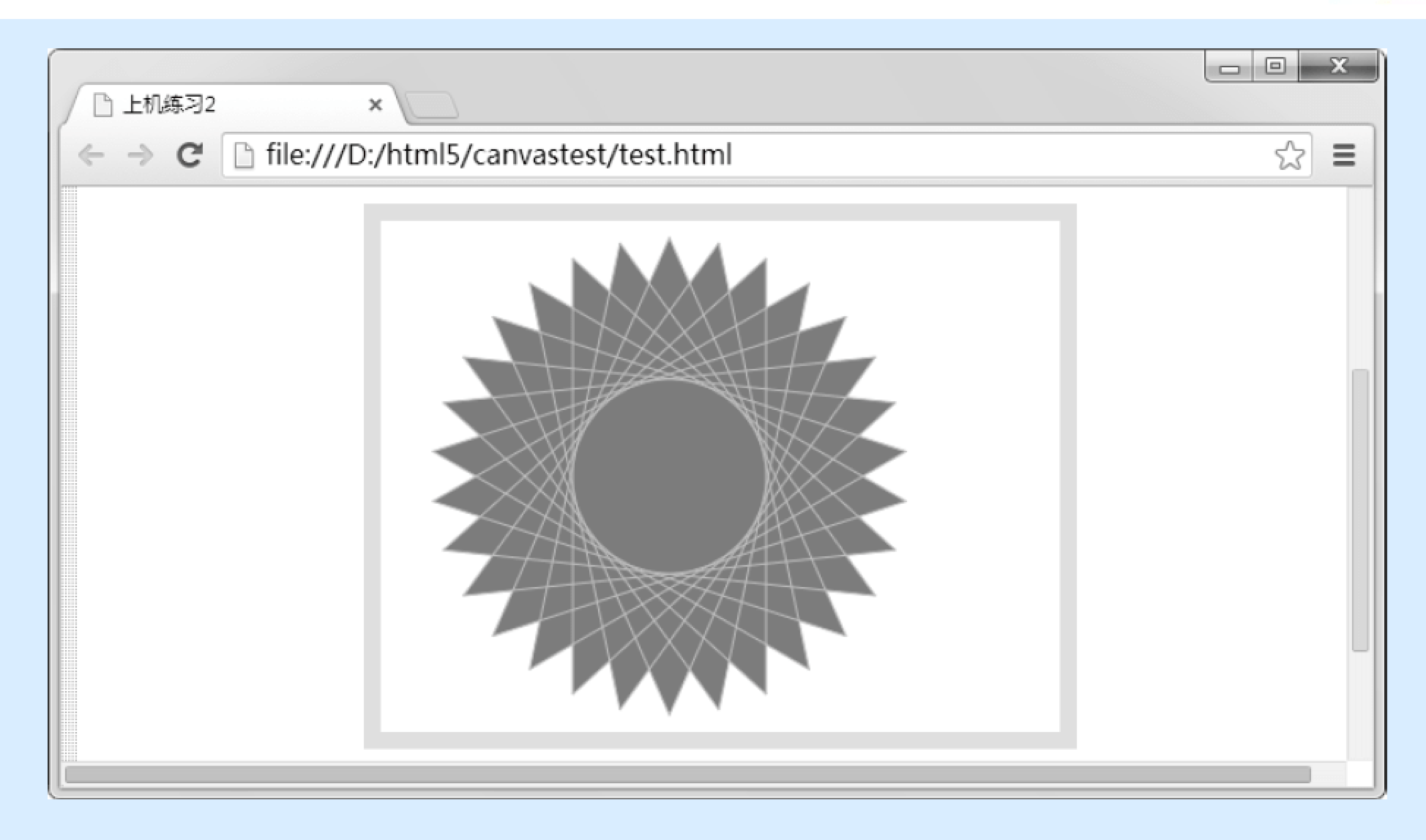


图 8-22　上机练习(2)的效果

(3)　对图像进行操作

利用 drawImage()方法的不同形式绘制图像，并且实现图像的平铺和裁剪功能，裁剪时的效果如图 8-23 所示。在该图中，左边是图片的原始效果，右侧是裁剪后的效果。

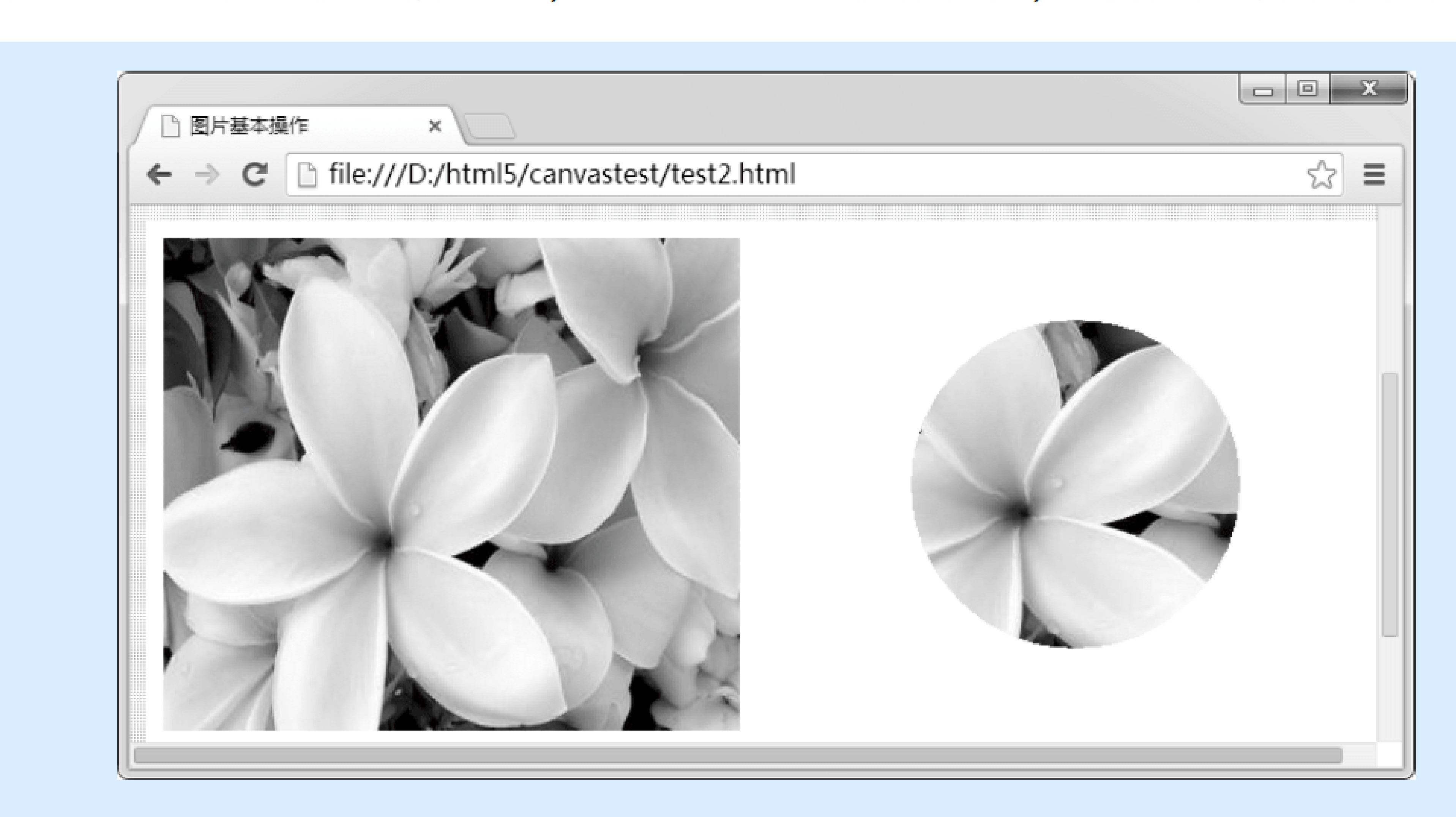


图 8-23　裁剪及其裁剪后的效果





第9章

HTML 5 的其他新特性

HTML 5 的出现越来越受到人们的重视与青睐，其中最大的原因就是解决了 HTML 4 的诸多局限性。在 HTML 5 中，这些局限性得到了很大程度的改善，像允许用户选择多个文件并读取其内容，能够在客户端存储大数据和使用数据库，甚至可以像窗体应用程序一样跨文档传递数据和实现多线程等。

本章将从 7 个方面介绍 HTML 5 的这些新增特性。通过本章的学习，读者可以了解 HTML 5 中提供的高级技术，也可以熟练使用这些技术实现某些常用的功能。

本章学习目标：

- 掌握 FileList 对象获取多个文件信息的方法
- 掌握 FileReader 接口的方法和事件
- 了解 FileError 接口的错误编码
- 熟悉 DataTransfer 对象的常用方法和属性
- 了解 localStorage 对象和 sessionStorage 对象的区别
- 掌握 JSON 的 parse()方法和 stringify()方法
- 熟悉 HTML 5 中多线程的实现过程
- 了解 HTML 5 的地图 API
- 了解 HTML 5 的离线缓存

9.1　文件新增特性

文件在 HTML 5 中占有非常重要的作用。HTML 5 为文件操作提供了一个全新的文件 API，通过这个 API，使得访问本地文件和处理文件信息变得十分简单。例如，可以同时选择多个文件，对文件扩展名进行过滤以及查看文件内容等。下面详细介绍 HTML 5 中文件 API 的具体操作。

9.1.1　获取多个文件的信息

HTML 5 同样可以使用 file 类型来创建一个文件域。具体方法是在 form 内创建一个 input 类型为 file 的元素，然后运行即可。浏览器会自动识别并创建相应的浏览按钮和选择文件对话框。

在 file 类型中选择的文件其实是一个 file 对象。file 对象有 4 个属性，如下所示。

- name：表示选中文件不带路径的名称。
- size：使用字节表示的文件大小。
- type：使用 MIME 类型表示的文件类型。
- lastModifiedDate：表示文件的最后修改日期。
- multiple：表示是否允许同时选择多个文件，默认值为 false。

【例 9.1】

创建一个示例，使用 HTML 5 的 file 对象获取用户选择文件的名称、大小、类型和修改日期。

步骤 01　首先需要创建一个表单，并将 file 对象的 multiple 属性设置为 true，使用户可以选择多个文件。代码如下：

```
<h2>上传照片</h2>
<form id="form1">
    选择文件
    <input type="file" id="bookfile" multiple="true" />
    <input type="button" value="提 交" onclick="selectedFiles()" />
</form>
```

步骤 02　在上传表单的下方添加一个表格，显示最终结果。代码如下：

```
<table width="100%" cellspacing="1" cellpadding="1" border="0"
  class="mytable">
    <tr>
        <th>文件名称</th>
        <th>文件大小</th>
        <th>文件类型</th>
        <th>上次修改日期</th>
    </tr>
    <tbody id="tFiles">
    </tbody>
</table>
```

步骤 03 现在运行示例，即可在弹出的对话框中选择多个文件。如图 9-1 所示为选中多个文件时的预览效果。

步骤 04 为了实现在单击“提交”按钮后显示这些文件的信息，还需要编写 selectedFiles()函数，具体实现代码如下：

```
function selectedFiles()
{
    var result = $("tFiles");
    for(var i=0; i<$("bookfile").files.length; i++)  //遍历选中的多个文件
    {
        var aFile = $("bookfile").files[i];
        var str = "<tr><td><img src=\"images/manico.gif\" />"
            + aFile.name + "</td><td>"
            + aFile.size + "字节</td><td>" + aFile.type + "</td><td>"
            + aFile.lastModifiedDate + "</td></tr>";
        result.innerHTML += str;
    }
}
```

当使用 multiple 属性后，用户选择的多个文件实际上保存在一个 files 数组中，其中的每个元素都是一个 file 对象。因此，为了获取每个文件的信息，需要对 files 数组进行遍历，再逐个获取文件名称、大小、类型和修改日期。代码运行效果如图 9-2 所示。



图 9-1 选中多个文件的效果



图 9-2 显示多个文件信息的效果

【例 9.2】

使用 file 对象的 type 属性可以获取文件类型，可以利用此属性来限制用户允许上传的文件类型。例如，希望只允许上传 JPG 图片，如果不是，则给出错误提示或跳过该文件。

实现方法是对示例中的 selectedFiles()函数进行修改，增加文件类型判断的功能，使用户只能上传图像类型的文件。如下所示是修改后的函数代码：

```
function selectedFiles()
{
    var result = $("tFiles");
    for(var i=0; i<$("bookfile").files.length; i++)
    {
        var aFile = $("bookfile").files[i];
```

```
        if(!/image\/\w+/.test(aFile.type))          //判断类型是否匹配
        {
            alert(aFile.name + "不是图像文件不能上传.");
            continue;
        }
        var str = "<tr><td><img src=\"images/manico.gif\" />"
            + aFile.name + "</td><td>"
            + aFile.size+"字节</td><td>" + aFile.type + "</td><td>"
            + aFile.lastModifiedDate + "</td></tr>";
        result.innerHTML += str;
    }
}
```

在这里，主要是通过判断 type 属性的值是否以“image/”开头来区分图像类型。现在运行程序，仍然可以在对话框中选择任何类型的文件，如图 9-3 所示为选择 5 个文件后的效果。

然后单击“提交”按钮，将会对文件类型进行判断，不匹配时将弹出对话框进行提示。最终仅在列表中显示符合条件的文件，如图 9-4 所示。



图 9-3　选择多个文件后的效果



图 9-4　提交后的运行效果

技巧：如果为 input 元素添加 accept 属性，也可指定要过滤的文件类型。例如，用 accept=“image/jpeg”同样可以实现例 9.2 的效果。

9.1.2　新增的FileReader接口简介

HTML 5 的 file 对象主要用于获取文件属性信息，如果要读取文件的内容，则需要调用 HTML 5 中新增的 FileReader 接口。FileReader 接口提供了一组异步 API，通过这些 API，可以从浏览器的主线程中异步访问文件系统。

表 9-1 中列出了 FileReader 接口中用于读取文件的方法及其说明。需要注意，无论读取成功或者失败，读取方法都不会返回读取结果，而是将结果保存在 result 属性中。

除了上述的方法之外，FileReader 接口还包含了一套完整的事件模型来监视读取文件时的各个状态。这些事件主要如表 9-2 所示。

表 9-1 FileReader 接口的方法

方法名称	说 明
readAsArrayBuffer()	使用字节数组格式读取文件内容
readAsText()	使用文本格式读取文件内容
readAsDataURL()	使用 URL 格式读取文件内容
abort()	中断当前读取操作

表 9-2 FileReader 接口的事件

事件名称	说 明
onabort	当读取数据中断时触发
onerror	当读取数据出错时触发
onloadstart	当读取数据开始时触发
onprogress	当正在读取数据时触发
onload	当读取数据成功时触发
onloadend	当读取操作完成时触发，无论成功或者失败都触发

【例 9.3】

由于并非所有浏览器都支持 HTML 5 的 FileReader 接口。因此，在使用之前必须先判断浏览器是否支持 FileReader 接口，代码如下所示：

```
if(typeof FileReader=="undefined"){
    alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本程序。");
}else{
    alert("您的浏览器环境正常。");
    var fd = new FileReader();
}
```

注意：当访问不同的文件时，必须创建不同的 FileReader 接口实例。因为每调用一次 FileReader 接口，都将返回一个新的 FileReader 对象，这样才能访问不同文件中的数据。

9.1.3 使用FileReader接口读取文件

FileReader 接口是 W3C 为读取文件内容在 HTML 5 中新增的，上节介绍了该接口的方法，下面详细介绍这些方法的具体应用。

1. 使用readAsText()方法

readAsText()方法可以以文本格式读取文件的内容。语法格式如下：

```
var result = FileReader.readAsText(blob, encoding);
```

第一个参数是 file 类型，表示要读取的文件，第二个参数是字符串类型，用于指定读取时使用的编码，默认值为 UTF-8。返回值为字符串形式的文件内容。

【例 9.4】

使用 readAsText()方法制作一个实现读取用户选择文本文件内容的案例，并最终将内容显示到页面上。

首先创建一个表单，添加文件上传域和结果显示布局。代码如下所示：

```
<h2>上传内容</h2>
<form id="form1">
    选择本地的文本文件
    <input type="file" id="bookfile" multiple="true" />
    <input type="button" value="读取" onclick="readFileContent()" />
</form>
<div class="text" id="result">
```

上述代码的重点是 file 类型和读取按钮，file 类型允许用户选择一个文件。单击“读取”按钮后，将执行 readFileContent()函数，将文件内容显示到下方的 div 中。

readFileContent()函数的实现代码如下所示：

```
function readFileContent()                     //读取文本文件的内容
{
    if($("bookfile").files.length)             //判断是否选择了文件
    {
        var aFile=$("bookfile").files[0];
        if(!/text\/\w+/.test(aFile.type))      //判断是否为文本文件
        {
            alert(aFile.name + "不是文本文件不能上传.");
            return false;
        }
        if(typeof FileReader=="undefined")   //判断当前浏览器是否支持FileReader接口
        {
           alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本程序。");
        }
        else
        {
            var fd = new FileReader();         //创建 FileReader 接口的对象
            fd.onload = function(res){         //显示文件内容
                $("result").innerHTML = this.result;
            }
            fd.readAsText(aFile);              //开始读取
        }
     }
     else
     {
         alert("没有选择文件，不能继续。");
         return false;
     }
}
```

如上述代码所示，在 readFileContent()函数中针对没有选择文件、文件类型不对以及浏览器不支持 FileReader 接口的情况进行了判断。真正使用 readAsText()方法读取文件内容

的代码非常简单，不过要注意结果属性 result 只能在 onload 事件中使用。

现在运行，即可查看读取文本文件内容的效果，如图 9-5 所示。



图 9-5　读取文本内容的效果

2. 使用readAsDataURL()方法

readAsDataURL()方法可以将文件读取为一串 URL 字符串。该字符串通常会使用特殊格式的 URL 形式直接读入页面，如图像格式等。该方法的语法格式如下：

```
var result = FileReader.readAsDataURL(blob);
```

blob 参数表示文件的只读原始数据对象。readAsDataURL()方法返回值是一个表示数据的本地对象。

【例 9.5】

使用 FileReader 接口的 readFAsDataURL()方法实现不通过后台即时实现图片预览的功能，其中允许用户选择多个图片文件，单击按钮提交后，显示这些文件的缩略效果图。

步骤 01 创建一个表单，将文件选择框的 multiple 属性设置为 true。然后向名称是“读取图像文件”的按钮中添加 onClick 事件属性。代码如下：

```
选择文件：<input type="file" id="selFile" multiple="true" /><br/><br/>
<input type="button" id="btnGetPicInfo" value="读取图像文件"
  onClick="GetPicFile()" />  
```

步骤 02 向页面中添加显示图片列表的 div 元素。代码如下：

```
<div id="result"></div>
```

步骤 03 GetPicFile()文件用于获取用户选择的文件，并且进行判断，如果选择的文件符合是图像的要求，则将其显示到页面中，否则会弹出提示。代码如下：

```
function GetPicFile()
{
    var filelist = document.getElementById("selFile");     //获取选择的文件
    if(filelist.files.length==0)                          //如果选择文件列表为空
    {
        alert("请选择您要查看的图像");
    }else{
        for (var i=0; i<filelist.files.length; i++) {  //循环显示文件
            var file = filelist.files[i];               //获取单个文件
            var imageType = /image.*/;              //声明文件类型
```

```
            if (!file.type.match(imageType)) {      //如果上传文件不合法
                alert(file.name + "不是图像文件，因此不能上传。");
                continue;
            }
            var reader = new FileReader();      //实例 FileReader 接口对象
            reader.onload = function(e){             //显示图像
                $("result").innerHTML += "<img src="
                    + e.target.result + " width=100 height=100 "
                    + "style='padding-right:1px;padding-bottom:1px;' />";
            };
            reader.readAsDataURL(file);
        }
    }
}
```

步骤 04 运行页面，查看效果，所有图像选择完成后单击“读取图像文件”按钮进行预览查看，效果如图 9-6 所示。



图 9-6　显示预览图像的效果

3. 使用readAsArrayBuffer()方法

readAsArrayBuffer()方法可以将文件读取为字节数组，通常是将它传送到后端，后端可以通过这段字符串存储文件。语法格式如下：

```
var result = FileReader.readAsArrayBuffer(blob);
```

blob 参数表示文件的只读原始数据对象。readAsArrayBuffer()方法返回值是一个表示数据的 ArrayBUffer 对象。readerAsBinaryString()方法的使用很简单，只要将一个要读取的文件 file 作为参数传入即可，这样读取的结果保存到 result 属性中。具体使用与 readAsDataURL()方法相同，这里就不再介绍了。

9.1.4 使用FileReader接口监听事件

在前面介绍 FileReader 接口中读取方法的使用时，仅对 onload 事件进行了处理。在读取正常的情况下，FileReader 接口中事件的触发顺序如下：

```
onloadstart -> onprogress -> onload -> onloadend
```

除了执行顺序外，在实际使用时还应该注意如下区别：

- 大部分的文件读取过程都集中在 onprogress 事件，该事件耗时最长。
- 如果文件在读取过程中出现异常或者中止，那么 onprogress 事件将结束，并直接触发 onerror 或者 onabort 事件，而不触发 onload 事件。
- onload 事件是文件读取成功时触发，而 onloaded 事件虽然也是文件操作成功时触发。但是 onloaded 事件无论读取成功或者失败，都会触发，这一点需要注意。因此，如果要正确获取文件数据，必须在 onload 事件中编写代码，而不是 onloaded 事件。



提示：FileReader 接口中的每个事件都表示读取文件时的一个状态。

【例 9.6】

创建一个示例来监听读取过程中的所有事件，并输出事件的执行先后顺序。布局代码如下所示：

```
<h2>上传图片</h2>
<form id="form1">
   选择文件：
   <input type="file" id="selFile" multiple="true" accept="image/jpeg" />
   <br/> <br/>
   <input type="button" id="btnGetPicInfo" value="读取图像文件"
     onClick="GetPicFile()" />
</form>
<div class="text" id="result">
   <img class="pic" id="bookpic" width="300px" height="200px"/>
   <ol id="eventList"></ol>
</div>
```

上述代码中 id 为 bookpic 的 img 元素用于显示选中的图片，id 为 eventList 的 ol 元素用于显示事件的执行顺序。

如下所示是单击“读取图像文件”按钮调用 GetPicFile()函数的实现代码：

```
function GetPicFile()
{
    var aFile = $("selFile").files[0];
    if(typeof FileReader=="undefined")
    {
        alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本程序。");
    }
    else
    {
        var fd = new FileReader();
        fd.readAsDataURL(aFile);
        //处理 onload 事件
        fd.onload=function(e){
            $("bookpic").src = this.result;
            $("eventList").innerHTML += "<li>触发了 onload 事件</li>";
        }
        //处理 onprogress 事件
        fd.onprogress = function(e){
```

```
            $("eventList").innerHTML += "<li>触发了 onprogress 事件</li>";
        }
        //处理 onabort 事件
        fd.onabort=function(e){
            $("eventList").innerHTML += "<li>触发了 onabort 事件</li>";
        }
        //处理 onerror 事件
        fd.onerror=function(e){
            $("eventList").innerHTML += "<li>触发了 onerror 事件</li>";
        }
        //处理 onloadstart 事件
        fd.onloadstart=function(e){
            $("eventList").innerHTML += "<li>触发了 onloadstart 事件</li>";
        }
        //处理 onloadend 事件
        fd.onloadend=function(e){
            $("eventList").innerHTML += "<li>触发了 onloadend 事件</li>";
        }
    }
}
```

上述代码对使用 readAsDataURL()方法读取图像过程中触发的所有事件进行了监听，并依次进行显示。最终通过显示的结果可以了解事件执行顺序。如图 9-7 所示为运行后的效果。



图 9-7　浏览器运行效果

9.1.5 文件读取时的异常处理

虽然使用 FileReader 接口中的方法可以快速实现对文件的读取。但是，在文件读取的过程中，不可避免地会出现各种类型的错误和异常。这时便可以通过 FileError 接口获取错误与异常所产生的错误代码，再根据返回的错误代码分析具体发生错误与异常的原因。

在出现下列情况时，可能会导致 FileReader 接口出现潜在的错误与异常：

- 访问某个文件的过程中，该文件被移动或者删除及被其他应用程序修改。
- 由于权限原因，无法读取文件的数据信息。

- 文件出于案例因素的考虑，在读取文件时返回一个无效的数据信息。
- 读取的文件太大，超出 URL 网址的限制，将无法返回一个有效的数据信息。
- 在读取文件的过程中，应用程序本身触发了中止读取的事件。

在异步读取文件的过程中，出现错误与异常都可以使用 FileError 接口。该接口主要用于异步提示错误，当 FileReader 对象无法返回数据时，将形成一个错误属性，而该属性则是一个 FileError 接口，通过该接口列出错误与异常的错误代码信息。

表 9-3 列出了 FileError 接口中提供的错误代码以及对应的说明。

表 9-3　FileError 接口的错误代码

错误代码	错误常量	说　明
1	NOT_FOUND_ERR	无法找到文件或者原文件已经被修改
2	SECURITY_ERR	由于安全考虑，无法读取文件数据
3	ABORT_ERR	由 abort 事件触发的中止读取过程
4	NOT_READABLE_ERR	由于权限原因，不能读取文件数据
5	ENCODING_ERR	读取的文件太大，超出读取时地址的限制

例如，下面的示例代码演示了处理读取文件时的异常处理过程：

```
var reader = new FileReader();
reader.readAsText(file,"gb2312");
reader.onload = function(e){                          //添加 onload 事件
    document.getElementById("showInfo").innerHTML = this.result;
}
reader.onerror=function(res){                         //添加 onerror 事件
    var num = res.target.error.code;                  //获取错误代码
    document.getElementById('showInfo').innerHTML = "文件无法显示：";
    if(num==1){
       document.getElementById('showInfo').innerHTML+="无法找到或原文件已被修改！";
    }else if(num==2){
       document.getElementById('showInfo').innerHTML+="无法获取数据文件！";
    }else if(num==3){
       document.getElementById('showInfo').innerHTML+="中止文件读取的过程！";
    }else if(num==4){
       document.getElementById('showInfo').innerHTML+="无权读取数据文件！";
    }else if(num==5){
        document.getElementById('showInfo').innerHTML+="读取的文件太大！";
    }
}
```

9.1.6 实战——实现文件上传

通过前面的练习，我们已经掌握了如何获取选择文件的基本信息，也能够限制文件的类型和读取文件的内容。但是这些文件只是保存在本地，并没有实现上传功能。

本次实战将使用 HTML 5 结合 JSP 实现将用户选择的多个文件批量上传到服务器。具体步骤如下。

步骤 01 使用 MyEclipse 工具创建一个名为 fileupload 的 Web 项目。

步骤 02 新建一个名为 index.jsp 的文件作为选择文件的页面。

步骤 03 在页面中创建一个表单，允许用户选择多个文件，并且可以查看要上传的图片文件。最终代码如下所示：

```
<h2>上传图片</h2>
<form action="upload" enctype="multipart/form-data" method="post">
  选择一张图片<input type="file" name="fileupload" value="upload"
multiple="true"  id="selFile" /><br> <input type="button" value="预览"
onClick="GetPicFile()" /><input type = "submit" value ="上传">
</form>
<div class="text" id="result"></div>
```

在这里要注意 form 的 enctype 属性值必须是 multipart/form-data，method 属性值必须是 post。

步骤 04 编写单击“预览”按钮时执行的 JavaScript 函数 GetPicFile()，具体代码可参考 9.1.3 节，这里不再重复。

步骤 05 在 fileupload 项目中新建一个名为 upload 的 Servlet。在 doPost()方法中编写代码，获取在客户端选择的多个文件，然后以流的形式写到项目下的 upload 目录中。代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse
  response) throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    out.println(
   "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");
    out.println("<HTML>");
    out.println(" <HEAD><TITLE>JSP 多文件上传后台程序</TITLE></HEAD>");
    out.println(" <BODY>");

    DiskFileItemFactory factory = new DiskFileItemFactory();
    String path = request.getSession().getServletContext()
                   .getRealPath("/") + "upload/";
    factory.setRepository(new File(path));
    factory.setSizeThreshold(1024 * 1024);
    ServletFileUpload upload = new ServletFileUpload(factory);
    try {
        List<FileItem> list =
           (List<FileItem>) upload.parseRequest(request);
        out.println("本次一次上传" + list.size() + "个文件。<br/>");
        out.println("<a href=\"index.jsp\">继续上传文件</a>");
        Iterator<FileItem> it = list.iterator();       // 遍历 list
        while (it.hasNext()) {
            FileItem fi = (FileItem)it.next();  // 类型转换
            System.out.println("该文件表单 name 为: " + fi.getFieldName());
            System.out.println("该文件文件名为: " + fi.getName());
            System.out.println("该文件文件类型为: " + fi.getContentType());
            System.out.println("该文件文件大小为: " + fi.getSize());
            if (fi.isFormField()) {// 判断该 FileItem 对象是否是一个普通表单类型
                String name = fi.getFieldName();    // 得到普通表单类型的表单名
```

```
                    String content = null;
                    try {
                        content = fi.getString("utf-8");
                    } catch (UnsupportedEncodingException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    } // 用指定编码得到普通表单的值
                    request.setAttribute(name, content); //键值放入 request 对象
                } else {
                    try {
                        // 得到文件表单的值，就是用户本地的文件路径
                        String pathStr = fi.getName();
                        // 如果文件表单为空，则不处理
                        if (pathStr.trim().equals("")) {
                            continue;
                        }
                        int start =
                         pathStr.lastIndexOf("\\");    //得到文件名在路径中的位置
                        String fileName =
                         pathStr.substring(start + 1); //得到文件名
                        File pathDest = new File(path, fileName);
                        System.out.println(fileName);
                        System.out.println("------------------------------");
                        fi.write(pathDest);              // 写文件
                        String name = fi.getFieldName();// 得到文件表单的名称
                        //把表单名、文件名放入 request
                        request.setAttribute(name, fileName);
                    } catch (Exception e) {
                        out.print("存储文件错误：");
                        e.printStackTrace();
                        return;
                    } finally { // 立即删除保存表单字段内容的临时文件
                        fi.delete();
                    }
                }
            }
        } catch (FileUploadException e) {
            e.printStackTrace();
        } catch (Exception e) {
             e.printStackTrace();
        }
        out.println("  </BODY>");
        out.println("</HTML>");
        out.flush();
        out.close();
}
```

注意，要正常运行上面的代码，还需导入两个 JAR 包，分别 commons-fileupload 和 commons-io。读者需要手动下载它们并放到项目的\WebRoot\WEB-INF\lib 目录中。

步骤 06 将 fileupload 项目部署到服务器。在浏览器中输入“http://localhost:8080/fileupload/”进入选择文件界面。如图 9-8 所示为选择了三个文件之后浏览器中的预览效果。



图 9-8　预览要上传的图片文件

步骤 07 单击“上传”按钮将把文件上传到 upload 目录，并且可以继续上传，上传结果如图 9-9 所示。

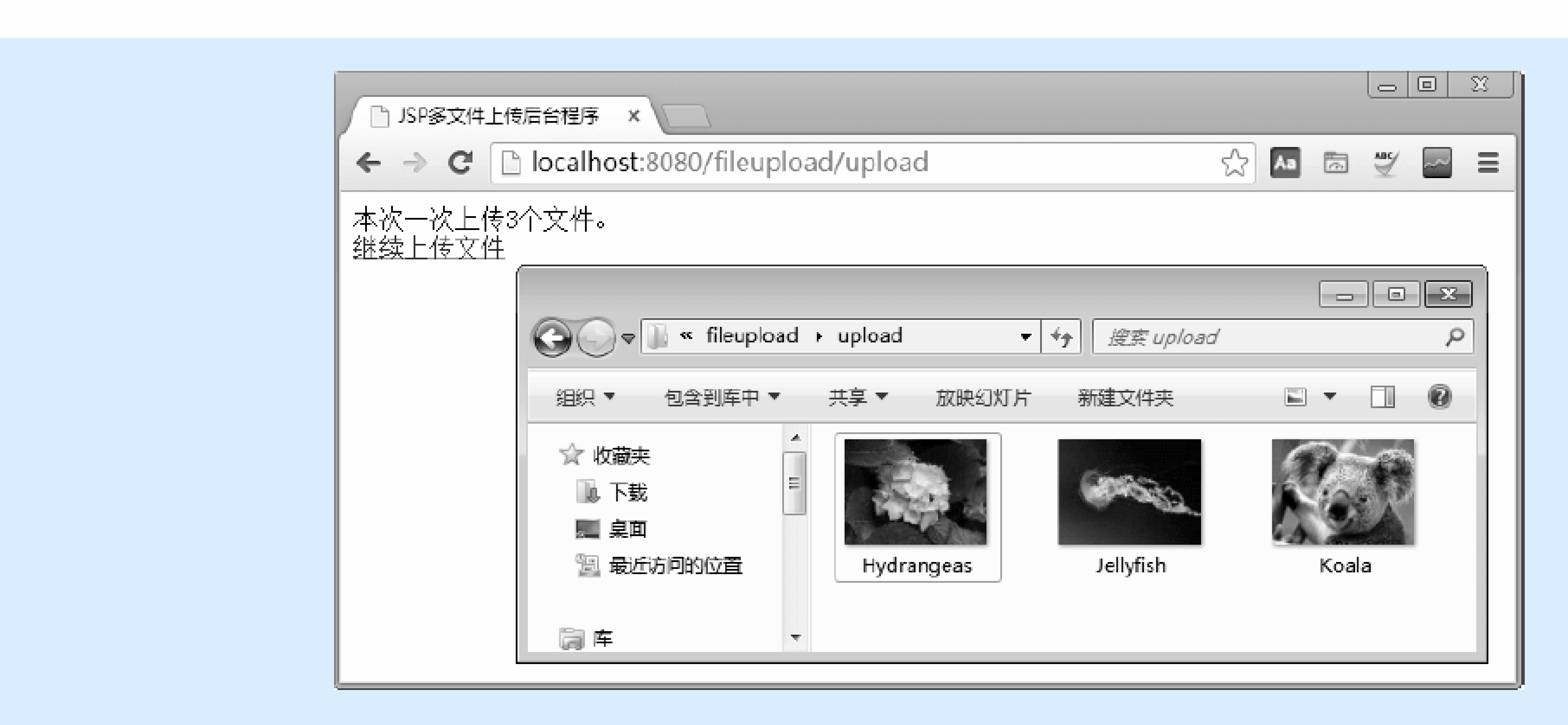


图 9-9　上传结果

9.2　拖 放 功 能

在 HTML 4 之前，如果要实现文件(例如文本文件、图像文件)的拖放功能，需要借助于 mousedown、mousemove 和 mouseup 事件实现在浏览器内部的拖放操作。而在 HTML 5 中提供了对文件拖放的支持，下面详细介绍 HTML 5 的拖放 API，以及拖放对象的使用。

9.2.1　拖放API简介

HTML 5 提供了文件拖放操作的 API，使用这些 API，不仅能够实现浏览器内部的拖放操作，还可以在浏览器与其他应用程序之间实现数据传输，同时大大简化了拖放的实现代码。

在实现拖放时需要涉及到很多事件，这些事件的说明如表 9-4 所示。

在实现拖放功能之前，还需要将想要拖放对象的 draggable 属性设置为 true，这表示允许该元素进行拖放。另外，img 元素和 a 元素(必须指定 href 属性)默认情况下允许拖放。

表 9-4 拖放事件及说明

事件名称	说 明
dragstart	开始执行拖放操作，应用于被拖放元素
drag	拖放过程中，应用于被拖放元素
dragenter	被拖放的元素开始进入本元素的范围内，应用于拖放过程中鼠标经过的元素
dragover	被拖放的元素正在本元素范围内移动，应用于拖放过程中鼠标经过的元素
dragleave	被拖放的元素离开本元素的范围，应用于拖放过程中鼠标经过的元素
drop	有其他元素被拖放到了本元素中，应用于拖放的目标元素
dragend	拖放操作结束，应用于拖放的对象元素

假设要将页面中 drag 元素拖放到 area 元素内，大致步骤如下。

步骤 01 在开始拖放元素时触发 drag 事件，示例代码如下所示：

```
document.getElementById("drag").addEventListener("dragstart", draging,
false);
```

步骤 02 在将拖放的源对象放到目标对象上面时触发 drop 事件，触发该事件之后调用 drop()函数，示例代码如下所示：

```
document.getElementById("area").addEventListener("drop", drop, false);
```

步骤 03 将拖放的源对象离开目标对象时触发 dragleave 事件，触发该事件之后调用 dragleave()函数，示例代码如下所示：

```
document.getElementById("area").addEventListener("dragleave", dragleave,
false);
```

步骤 04 添加页面的 ondragover 事件，示例代码如下：

```
document.ondragover = function (e) { e.preventDefault(); }
```

步骤 05 添加页面的 drop 事件。创建拖拽事件监听的时候把默认的行为事件禁用，因为浏览器默认有拖拽行为。HTML 5 中 dragover 事件要使用 e.preventDefault()禁用阻止的方法，不然 drop 事件可能不会被触发。示例代码如下：

```
document.ondrop = function (e) { e.preventDefault(); }
```

9.2.2 拖放对象的方法和属性

一般情况下拖动某个元素，然后把它拖放到另外某个元素上也需要在源元素与目标元素之间传送一些数据。

为了完成这项数据传送任务，HTML 5 提供了拖放对象——DataTransfer。

DataTransfer 对象提供了对剪贴板的访问，以便在拖放操作中使用。由于 DataTransfer 是事件对象的属性，所以只能在拖放事件的事件处理程序中访问 DataTransfer 对象。

DataTransfer 对象既可以在源对象中使用，也可以在目标对象中使用。DataTransfer 对

象有多个方法，常用方法及说明如表 9-5 所示。

表 9-5　DataTransfer 对象的方法说明

方法名称	说　明
getData()	获取由 setData()中保存的数据并且进行返回。如果返回的数据不存在，则返回空字符串
setData()	为元素添加指定的数据。有两个参数，第一个参数是一个字符串，表示保存的数据类型，也是 getData()方法唯一的一个参数；第二个参数表示数据
clearData()	清除以特定格式保存的数据。如果未指定格式，则删除当前元素的所有携带数据
addElement()	为拖动操作添加一个元素。添加这个元素只影响数据(即增加作为拖动源而影响回调的对象)，不会影响拖动操作时页面元素的外观
setDragImage()	指定一幅图像，当拖动发生时，显示在光标下方。该方法有 3 个参数，它们分别是要显示的 HTML 元素和光标在图像中的 x、y 坐标。其中，HTML 元素可以是一张图像，也可以是其他元素

其中 getData()和 setData()方法最为常用，setData()方法用于源事件，以便提供关于将要进行传送的数据。相应地，getData()方法用于目标事件，确保获取的数据和数据格式。

例如，可以通过 setData()方法设置一个文本数据，并且通过 getData()方法来接收数据。代码如下：

```
event.dataTransfer.setData("text", "this is a text data");
var text = event.dataTransfer.getData("text");
```

DataTransfer 对象的常用属性及说明如表 9-6 所示。

表 9-6　DataTransfer 对象的属性说明

属性名称	说　明
types	返回在 dragstart 事件出发时为元素存储数据的格式，如果是外部文件的拖放，则返回 files
dropEffect	返回已选择的拖放效果，即能够获取被拖放的元素能够执行哪种放置行为
effectAllowed	返回允许执行的拖放操作效果，即表示允许拖放元素的哪种 dropEffect

要注意 dropEffect 属性只有与 effectAllowed 属性搭配才有用，如果它所设置的效果与 effectAllowed 属性设置的效果不符，则拖放操作失败。dropEffect 属性可有如下取值。

- none：不能把拖动的元素放在这里。这是除文本框之外所有元素的默认值。
- move：应该把拖动的元素移动到放置目标。
- copy：应该把拖动的元素复制到放置目标。
- link：表示放置目标会打开拖动的元素(但拖动的元素必须是一个链接，存在 URL)。

与 dropEffect 属性相比，effectAllowed 属性的取值较多，具体说明如下。

- none：被拖动的元素不能有任何行为。

- move：只允许值为 movedropEffect。
- copy：只允许值为 copy 的 dropEffect。
- link：只允许值为 link 的 dropEffect。
- linkMove：允许值为 link 和 move 的 dropEffect。
- copyMove：允许值为 copy 和 link 的 dropEffect。
- copyLink：允许值为 copy 和 link 的 dropEffect。
- uninitialized：没有该被拖动元素放置行为。
- all：允许任意 dropEffect。

注意：如果要使用 dropEffect 属性，必须在 ondraggenter 事件处理程序中针对目标元素进行设置；如果要使用 effectAllowed 属性，必须在 ondraggstart 事件处理程序中进行设置。

如下代码展示了属性的简单使用：

```
source.addEventListener("dragstart", function(ev){
    var dt = ev.dataTransfer;                  //向 dataTransfer 对象中追加数据
    dt.effectAllowed = "all";
    dt.setData("text", "拖动文本");              //设置拖动元素的值
}, false);
dest.addEventListener("dragend", function(ev){
    ev.preventDefault();                       //不执行默认处理(拒绝被拖放)
})
```

dropEffect 属性与 effectAllowed 属性的值设置必须合法，下面总结了一些常用的规则：

- 如果 effectAllowed 属性设置为 none，则不允许拖放要拖放的元素。
- 如果 dropEffect 属性设置为 none，则不允许被拖放到目标元素中。
- 如果 effectAllowed 属性设置为 all 或不设置，则 dropEffect 属性允许被设置为任何值，并且按其指定的值进行显示。
- 如果 effectAllowed 属性设置为具体效果值(不为 none 和 all)时，dropEffect 属性也设置了具体效果值，则两个值必须完全相等，否则不允许被拖放元素拖放到目标元素中。

9.2.3 实战——模拟图片删除

本项目案例通过拖放 API 将相册中的某张图片以拖动的方式放入回收站，从而实现相册管理功能。其具体步骤如下。

步骤 01 添加新的 HTML 页面，在页面的合适位置添加 3 个 div 元素，它们分别表示图片列表、显示删除图片的效果以及显示回收站。然后向第一个 div 元素中添加 ul 和 li 元素，它们实现显示多张图片的效果。页面的具体代码如下：

```
<body onLoad="init();">
  <h2>相册管理</h2>
    <div class="text" id="result">
    <div id="picList">
```

```
<ul>
    <li id="li01"><img src="images/1.jpg" draggable="true" /></li>
    <li id="li02"><img src="images/2.jpg" draggable="true" /></li>
    <li id="li03"><img src="images/3.jpg" draggable="true" /></li>
    <li id="li04"><img src="images/4.jpg" draggable="true" /></li>
</ul>
</div>
<div style="clear:both"></div>
<div id="status" style="display:none;"><p id="pStatus" class="del"></p>
</div>
<div id="laji"></div>
</div>
</body>
```

步骤 02 页面加载调用自定义的 init()函数，该函数的具体代码如下：

```
var intDeleNum = 0;                              //声明删除图片的变量
function init(){
    var drag = document.getElementsByTagName("li");     //获取所有的 li 元素
    for(var i=0; i<drag.length; i++){                   //遍历该元素
        drag[i].addEventListener("dragstart",function(e){  //添加 dragstart 事件
            var dt = e.dataTransfer;                    //获取 dataTransfer 事件
            dt.setData("text/plain",this.id);           //向对象中存入数据
        },false);
    }
    var recy = document.getElementById("laji");         //获取目标元素
    recy.addEventListener("drop",function(e){           //添加 drop 事件
        var dt = e.dataTransfer;                        //获取 dataTransfer 对象
        var intval = dt.getData("text/plain");          //从对象中获取数据
        Drop(intval);                                   //删除图片
    },false);
}
```

上述代码中先声明全局变量 intDeleNum，该变量保存删除图片的数量。接着在 init()函数中首先通过 getElementsByTagName()方法获取全部相册中的元素，然后遍历全部元素时为每一张图片元素添加拖动时触发的 dragstart 事件。在该事件中调用 dataTransfer 对象存入每一张图片元素对应的 ID 号，即 this.id 的值。然后获取 ID 为 laji 的目标元素，向该元素添加 drop 事件，在该事件中调用 getData()函数读取存入的单个图片元素的 ID 号，将该 ID 号作为实参调用 Drop()函数移除图片。

步骤 03 Drop()函数中通过 removeChild()方法移除相册中指定的图片，形成删除的效果，同时通过全局变量 intDeleNum 累计删除图片的数量，并将该总数量的值显示到页面中。该函数的具体代码如下：

```
function Drop(id)
{
    var node = document.getElementById(id);         //获取指定的 li 元素
    node.parentNode.removeChild(node);              //移除图片
    intDeleNum++;                                   //累计删除数量
    document.getElementById("status").style.display = "block";
    document.getElementById("pStatus").innerHTML =
        "已经成功删除" + intDeleNum + "张图片";
}
```

步骤 04 通过 document 对象添加页面的 dragover 事件和 drop 事件，它们都调用 e.preventDefault()方法取消页面的默认值。具体代码如下：

```
document.ondragover = function(e)
{
    e.preventDefault();                //阻止默认方法，取消拒绝被拖放
}
document.ondrop = function(e)
{
    e.preventDefault();                //阻止默认方法，取消拒绝被拖放
}
```

上述代码中添加页面的 dragover 事件和 drop 事件，它们都使用 e.preventDefault()方法取消页面的默认值，允许拖放页面。

步骤 05 为页面中的元素添加样式效果，其主要的样式代码如下所示：

```
#picList img{
    width: 100px;
    height: 100px;
}
#picList ul li{
    float: left;
    list-style-type: none;
    list-style: none;
    margin: 3px;
}
#laji{
    background-image: url(images/5.gif);
    border: 1px solid gray;
    width: 203px;
    height: 254px;
    margin: auto;
    text-align: center;
}
```

步骤 06 运行本案例的代码进行测试，拖动删除图片时的效果如图 9-10 所示。删除图片成功时的效果如图 9-11 所示。

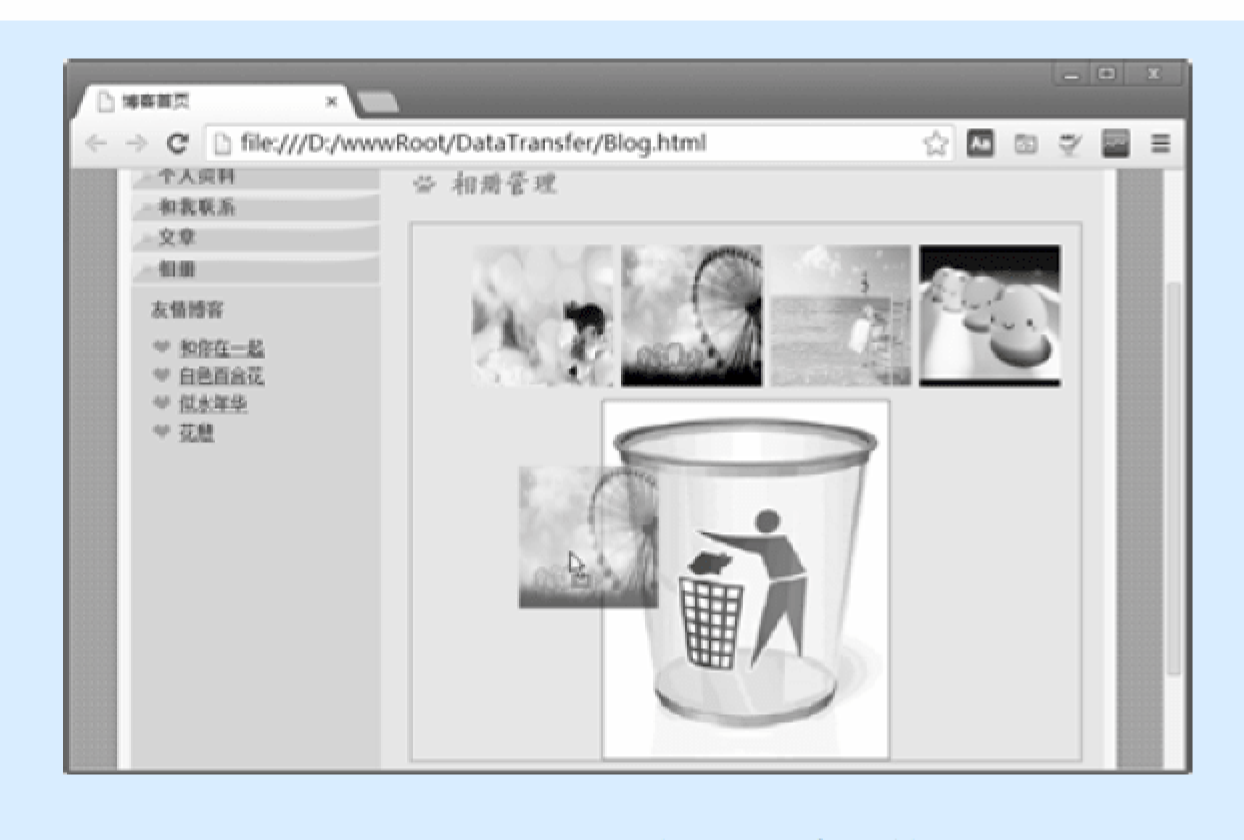


图 9-10　拖动删除图片时的效果

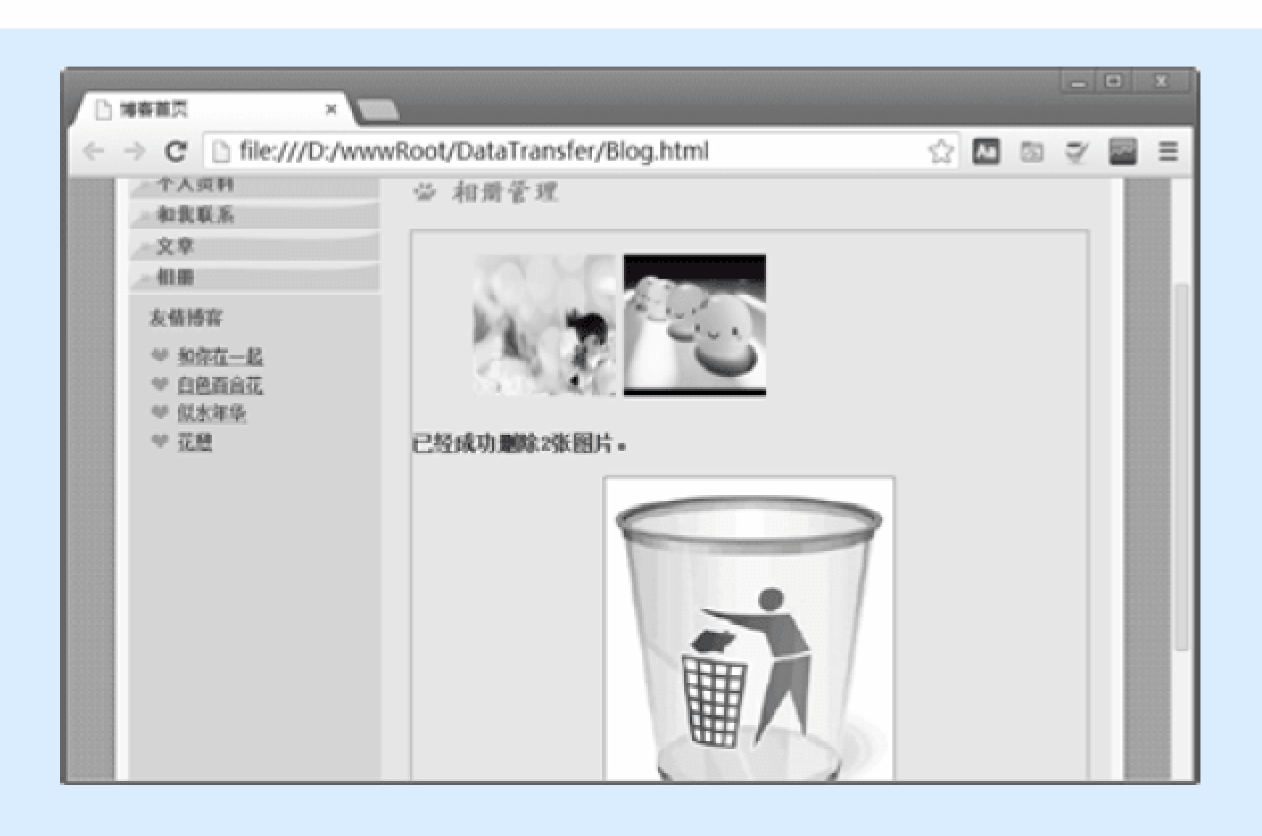


图 9-11　删除图片成功时的效果

9.3　新增的客户端数据存储特性

随着 Web 应用的发展，应用开发者越来越关注的一个问题是：如何更好地在客户端存储数据。在 HTML 4 中通常使用 Cookie 存储机制，但是由于 cookies 有限制保存数据空间大小、数据保密性差、代码操作复杂等缺点，已经完全无法满足如今开发者的需求。

HTML 5 规范定义了一种在客户端存储数据的更好的方式(即 Web 存储)，本节将介绍这种方式。

它包含两种不同的存储对象类型：sessionStorage 和 localStorage。这一节我们就主要来介绍 sessionStorage 对象和 localStorage 对象。

9.3.1　客户端存储对象简介

HTML 5 的客户端存储是在 Cookie 基础上发展起来的，它与 Cookie 相比，具有如下优势。

- 存储空间更大：默认情况下可以存储 5MB 大小的数据，虽然不同浏览器支持的数据大小会有所不同，但是都要比 Cookie 大得多。
- 存储内容不会发送到服务器：Cookie 的内容会随着请求一起发送到服务器，这对于本地存储的数据是一种带宽浪费。而客户端存储中的数据仅仅是保存在本地，不会与服务器发生任何交互。
- 更多丰富易用的接口：客户端存储提供了一套更为丰富的接口，使得数据操作更为简便。

HTML 5 的客户端存储机制主要由两大对象组成，即 localStorage 和 sessionStorage。其中 sessionStorage 用于存储与用户会话有关的数据，作用周期与会话相同；localStorage 用于持久化的存储数据，除非手动删除数据，否则数据永远不会过期。

1. sessionStorage对象

sessionStorage 对象所存储的数据只有在同一个会话中的页面才能访问，并且当会话结

束后，数据也随之销毁。因此，sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。

sessionStorage 对象最常用的方法如下所示。

- setItem(key,value)：参数 key 表示被保存内容的键，value 表示被保存内容的值。
- getItem(key)：获取指定 key 本地存储的值，如果不存在，则返回一个 null 值。
- removeItem(key)：删除指定 key 本地存储的值。
- clear()：清除 localStorage 对象中所有的数据。
- key()：获取指定下标的键名称(如同 Array)。

2. localStorage对象

使用 sessionStorage 对象保存的用户会话数据在关闭浏览器之后会丢失。因此，如果需要在客户端保存永久数据，则应该使用 localStorage 对象。localStorage 对象可以将数据长期保存在客户端，直至手动清除为止。

localStorage 对象同 sessionStorage 对象一样，最常用的方法有 setItem()、getItem()、removeItem()、clear()和 key()，在这里就不再详细介绍这些方法。

9.3.2 操作本地数据

由于 localStorage 对象和 sessionStorage 对象的方法是一样的，本节以 localStorage 对象为例讲解如何写入数据、读取数据以及清空数据。

1. 判断浏览器是否支持

使用 localStorage 对象之前，需要判断浏览器是否支持该对象，如果不支持，它可能会为空。判断代码如下：

```
function getLocalStorage() {
    try {
        if(!!window.localStorage)       //如果浏览器支持，则返回该对象
            return window.localStorage;
    } catch(e) {
        return undefined;
    }
}
```

2. 保存数据

要向客户端中保存一个数据，需要调用 setItem()方法，其调用格式如下所示：

```
localStorage.setItem(key,value)
```

其中，参数 key 表示数据的键名，参数 value 表示要保存的数据。一旦键名设置成功，则不允许修改，也不能重复；如果键名重复，那么将用新的键值取代原有的键值。

【例 9.7】

创建一个示例，使用 localStorage 对象将网站的名称保存到客户端。代码如下所示：

```
var localStorage = getLocalStorage();  //检测浏览器是否支持 localStorage 对象
if(localStorage){
    localStorage.setItem("sitename","窗内网");   //存储网站名称
}else{
    alert("您的浏览器不支持 localStorage 对象");
}
```

上述代码使用 localStorage 对象的 setItem()方法向客户端写入一个键名为 sitename，值为“窗内网”的数据。

除了 setItem()方法，也可以使用 localStorage[key]=value 或者 localStorage.key=value 形式写入数据。例如，上面的语也可写成如下形式：

```
localStorage["sitename"] = "窗内网";
localStorage.sitename = "窗内网";
```

3. 读取数据

如果要读取被保存的数据，可以调用 localStorage 对象中 getItem()方法，其调用格式如下：

```
localStorage.getItem(key);
```

其中，参数 key 表示数据保存时的键名，该方法将返回一个指定键名对应的键值；如果不存在，则返回一个 null 值。

例如要读取上面保存的网站名称，可用如下代码：

```
var sitename = localStorage.getItem("sitename");        //获取网站名称
```

注意：尽管使用 localStorage 对象可以将数据长期保存在客户端，但在跨浏览器读取数据时，被保存的数据也不可共享。即每一个浏览器只能读取各自浏览器中保存的数据，不能访问其他浏览器中保存的数据。

4. 清空数据

调用 localStorage 对象中的 removeItem()方法可以删除某个键名对应的数据。但是，有时保存数据很多，如果使用 removeItem()方法逐条删除比较麻烦。此时，可以调用 localStorage 对象中的另一个方法 clear()，该方法的功能是清空全部 localStorage 对象保存的数据，其调用格式如下所示：

```
localStorage.clear();
```

clear()方法是一个无参数方法，表示清空全部的数据。

5. 遍历数据

使用遍历可以查看 localStorage 对象中保存的全部数据信息。在遍历过程中需要借助 localStorage 对象的 length 属性和 key 属性，属性说明如下。

- length：表示 localStorage 对象中保存数据的总量。

- key：表示保存数据时的键名。该属性常与索引(index)配合使用，表示第几条键名对应的数据记录。其中，索引号以 0 值开始，例如第 2 条键名对应的数据 index 值应该为 1。

【例 9.8】

使用 localStorage 对象实现一个收藏夹的保存、查看和清空功能。具体步骤如下。

步骤 01 新建一个 HTML 页面。在页面中添加输入网站名称和地址的文本框，以及保存和清除按钮，代码如下：

```
<h2>管理收藏夹</h2>
网站名称：<input name="sitename" type="text" id="sitename" /><br/>
网站地址：<input name="siteurl" type="text" id="siteurl" /> <br/>
<input type="botton" value="保存" onclick="SaveData()"/>
<input type="botton" value="全部清除"  onclick="ClearData()"/>
```

步骤 02 使用 table 元素制作一个表格来显示当前收藏夹的内容。如下所示：

```
<table width="100%" cellspacing="1" cellpadding="1" border="0"
  class="mytable">
  <tr>
    <th>编号</th>
    <th>网站名称</th>
    <th>网站 URL 地址</th>
  </tr>
  <tbody id="list">  </tbody>
</table>
```

步骤 03 输入要保存的网站名称和地址之后单击“保存”按钮，将触发 SaveData()函数。SaveData()函数的实现代码如下：

```
function SaveData()
{
  var localStorage = getLocalStorage();          //创建 localStorage 对象
  if(localStorage){
     var id = $("sitename").value;               //获取网站名称
     var url = $("siteurl").value;               //获取网站地址
     localStorage.setItem(id,url);               //保存数据
     loadAllData();                              //加载数据
  }else{
    alert("您的浏览器不支持 localStorage 对象");
  }
}
```

上述代码首先创建一个 localStorage 对象，然后判断该对象是否为空。如果不为空再使用网站名称作为键，网站地址作为值保存到客户端。

步骤 04 loadAllData()函数的作用是从 localStorage 对象中遍历所有的收藏夹并显示到页面上的表格中。具体代码如下：

```
function loadAllData()
{
    var localStorage = getLocalStorage();
    var strHTML = "";
    for(var i=0; i<localStorage.length; i++){           //遍历所有数据
```

```
        var strkey = localStorage.key(i);        //根据索引获取键名，即网站名称
        //根据键名获取数据值，即网站地址
        var strval = localStorage.getItem(strkey);

        strHTML += "<tr><td><img src=\"images/manico.gif\" />"
                    + (i+1) + "</td>";
        strHTML += "<td>" + strkey + "</td>";
        strHTML += "<td>" + strval + "</td>";
        strHTML += "</tr>";
    }
    $("list").innerHTML = strHTML;                //追加到网页
}
```

步骤 05 创建 ClearData()函数，调用 localStorage 对象的 clear()方法清空所有数据。代码如下：

```
function ClearData()
{
    var localStorage = getLocalStorage();
    localStorage.clear();
    loadAllData();
}
```

步骤 06 如上面步骤所示，除了在添加和清空数据时需要更新页面之外，还需要在页面加载完成后更新数据。代码如下所示：

```
window.onload = loadAllData;
```

步骤 07 在浏览器中运行 HTML 页面，第一次加载时列表为空，界面效果如图 9-12 所示。



图 9-12　初始界面

步骤 08 当输入网站名称和网站地址并单击“保存”按钮之后，下面的表格会立即显示出来。如图 9-13 所示为添加 3 个网站时的运行效果。

步骤 09 在 Chrome 浏览器中按 F12 键，从 Resources 选项卡下单击 Local Storage 标签，可以查看 loccalStorage 对象在本地保存的数据。此时会看到与界面上显示的一致，如图 9-14 所示。



图 9-13　保存数据后的界面



图 9-14　查看保存的数据

注意：各浏览器查看 localStorage 对象所保存的数据方式不完全相同，Firefox 使用 Firebug 调试工具作为存储查看器。

步骤 10　最后单击“全部清除”按钮，所有数据都会被删除，如图 9-15 所示。

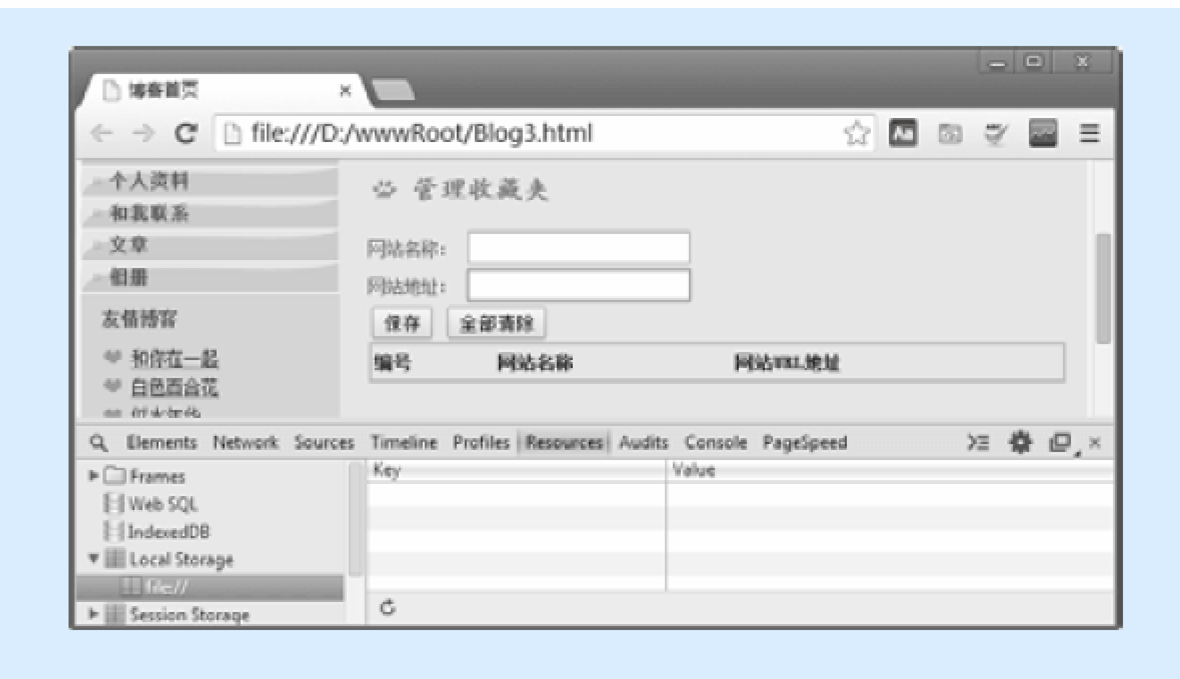


图 9-15　删除数据

9.3.3　实战——以JSON方式存取数据

无论 localStorage 对象还是 sessionStorage 对象，都是通过键值对来存储数据的。因此它们实现的只是支持字符串到字符串的映射，如果要存储很多的键值内容，它们的问题就出现了：处理相对复杂，扩展性差，数据结构不合理等。

为了解决这一问题，HTML 5 中还可以通过使用 JSON 对象转换数据，实现存储更多数据的功能。JSON 对象的常用方法如下。

- JSON.parse(data)：参数 data 表示 localStorage 对象获取的数据，调用该方法返回一个装载 data 数据的 JSON 对象。
- JSON.stringify(obj)：参数 obj 表示一个任意的实体对象，调用该方法返回一个由实体对象转成 JSON 格式的文本数据集。

下面通过具体的案例讲解 JSON 方式的数据保存与读取。具体步骤如下。

步骤 01 创建一个 HTML 页面。编写用于输入网站收藏夹信息的布局代码，并且可以根据网站名称进行搜索，以及显示搜索结果的表格。代码如下所示：

```
<h2>管理收藏夹</h2>
网站编号：<input name="sitename" type="text" id="siteid" />    <br/>
网站名称：<input name="sitename" type="text" id="sitename" />    <br/>
网站地址：<input name="siteurl" type="text" id="siteurl" />    <br/>
网站备注：<input name="siteurl" type="text" id="sitetext" />    <br/>
<input name="按钮" type="button" value="保存" onclick="SaveData()"/> <br/>
根据网站名称搜索:<input type="text" id="namekey">
<input type="button" value="检索" onclick="findStorage();" />  <hr/>
<div class="text">
    <table width="100%" cellspacing="1" cellpadding="1" border="0"
      class="mytable">
        <tr>
          <th>编号</th>
          <th>网站名称</th>
          <th>网站 URL 地址</th><th>备注</th>
        </tr>
        <tbody id="list">
        </tbody>
    </table>
```

步骤 02 在页面中可以输入网站编号、名称、地址和备注，单击“保存”按钮之后，将以 JSON 形式保存到客户端。实现代码如下：

```
function SaveData(){
    var localStorage = getLocalStorage();
    var data = new Object;                        //创建 JSON 对象
    data.id = $("siteid").value;
    data.name = $("sitename").value;
    data.url = $("siteurl").value;
    data.text = $("sitetext").value;
    var str = JSON.stringify(data);
    localStorage.setItem(data.name,str);       //保存数据
    alert("数据已保存");
}
```

在上述代码中，使用 JSON.stringify()方法把对象转换成 JSON 格式的文本数据，然后调用 setItem()方法保存 JSON 转换的文本数据。

步骤 03 当保存了多个 JSON 对象数据之后，可以通过网站名称作为关键字查找详细信息。如下所示为“检索”按钮执行的 findStorage()函数代码：

```
function findStorage(){
    var localStorage = getLocalStorage();
    var key = $("namekey").value;
    var siteObject = localStorage.getItem(key);
    var data = JSON.parse(siteObject);
    var strHTML =
      "<tr><td><img src=\"images/manico.gif\" />" + data.id + "</td>";
    strHTML += "<td>" + data.name + "</td>";
    strHTML += "<td>" + data.url + "</td>";
    strHTML += "<td>" + data.text + "</td>";
    strHTML += "</tr>";
    $("list").innerHTML = strHTML;
}
```

findStorage()函数使用网站名称作为键，调用 getItem()方法获得它的数据值。再调用 JSON 的 parse()方法将 localStorage 中获取的数据转换成 JSON 对象，最后使用“data.对象名”重新赋值。

步骤 04　在浏览器中运行页面，向客户端保存一些网站信息。然后输入其中一个网站的名称，再单击“检索”按钮查看结果，如图 9-16 所示。

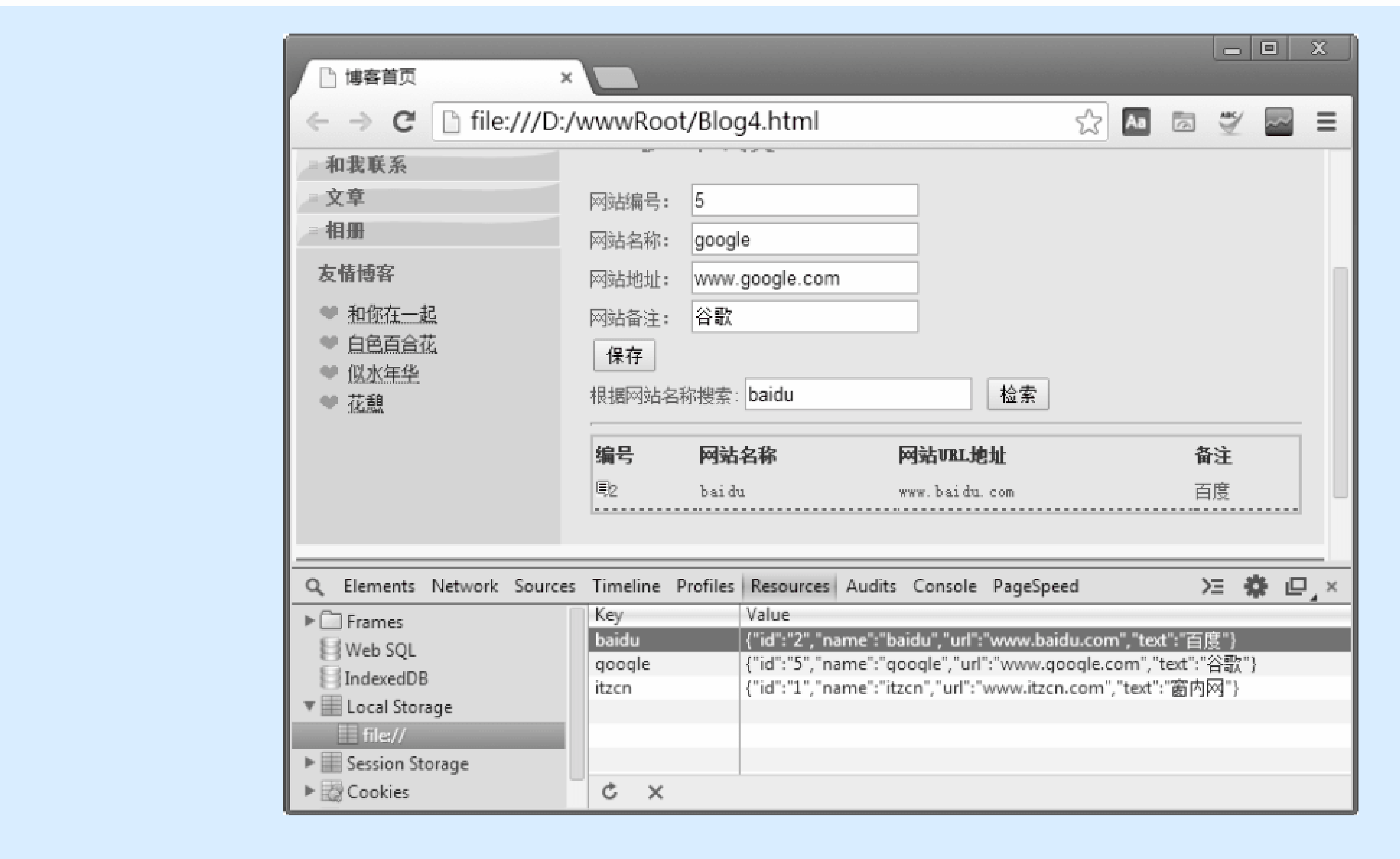


图 9-16　使用JSON读取数据

9.4　新增的本地数据库特性

localStorage 对象和 sessionStorage 对象虽然提供了比 Cookie 更好、更快和容量更大的存储数据，但是，如果数据之间的关系比较复杂，并且存储的数据容量比较大，这些空间往往也不能够满足用户的需要。这时可以使用 HTML 5 的另一个新特性——本地数据库。

所谓本地数据库，就是 HTML 5 内置的 SQLite 数据库的功能，能在客户端实现关系数据库的功能，下面详细介绍其应用。

9.4.1 HTML 5 本地数据库简介

在 HTML 4 以及之前的版本中，数据库只能存放在服务器端，通过动态语言来访问数据库。但是在 HTML 5 中，可以像本地文件那样轻松地对内置数据库进行直接访问。

HTML 5 本地数据库全称是 Web SQL DataBase，它提供了关系数据库的基本功能，可以存储页面中交互的、复杂的数据。它既可以保存数据，也能缓存从服务器获取的数据。HTML 5 本地数据库通过事务驱动，实现对数据的管理。因此，它可以支持多浏览器的并发操作，而不发生存储时的冲突。

与客户端数据存储相比，HTML 5 本地数据库具有如下优势：

- 可以自定义要设置的存储空间。
- 可以跨域访问。
- 存储结构更加自由。
- 可以方便地使用 Web SQL 进行读写。

要通过 HTML 5 本地数据库存储数据，第一步就是创建或打开一个数据库。这需要调用 openDatabase()方法，该方法的语法格式如下：

```
openDatabase(DBName,DBVersion,DBDescribe,DBSize,Callback());
```

其中，参数 DBName 表示数据库名称；参数 DBVersion 表示版本号；参数 DBDescribe 表示对数据库的描述；参数 DBSize 表示数据库的大小，单位为字节。如果是 2MB，必须写成 2*1024*1024；参数 Callback()表示创建或打开数据库成功后执行的一个回调函数。

调用 openDatabase()方法后，如果指定的数据库名存在，则打开该数据库；否则新创建一个指定名称的空数据库。

例如，下面的示例代码演示了如何创建一个数据库：

```
function CreateDatabase(){
    var db;
    db = openDatabase('studentDB','2.0','stumanager',2*1024*1024,
    function(){
        alert("数据库创建成功");
    });
}
```

上述代码创建了一个名为 studentDB，版本号为 2.0 的 2MB 数据库对象；如果创建成功，则执行回调函数，在回调函数中显示执行成功的提示信息。

9.4.2 数据库操作API

在创建和打开数据库之后便可以使用 HTML 5 本地数据库提供的 API 对数据进行管理。其中最常用的是 transaction()方法和 executeSql()方法。

1. transaction()方法

transaction()方法用来执行事务处理。使用事务处理可以防止在对数据库进行访问及执行有关操作的时候受到外界打扰。因为在 Web 上，同时会有许多用户都在对页面进行访问。在访问数据库的过程中，如果正在操作的数据被别的用户给修改的话，也会引起很多意想不到的后果。因此，可以使用事务来达到在操作完成之前，阻止其他用户访问数据库的目的。

transaction()的使用非常简单，直接通过返回的数据库对象调用即可。基本语法如下：

```
transaction(callbackFun,errorCallbackFun,successCallbackFun);
```

在上述语法形式中，可以传入 3 个参数，具体说明如下。

- callbackFun：回调函数，在该函数中执行访问数据库的 SQL 语句操作。
- errorCallbackFun：发生错误时调用的回调函数。
- successCallbackFun：执行成功时的回调函数。

一般情况下，使用 transaction()时传入一个类型事务的函数参数 tx 最为常用，它可以创建表或者执行操作语句。如下代码创建了一个数据库表：

```
db.transaction(function(tx){
    tx.executeSql("CREATE TABLE IF NOT EXISTS User (id unique, text)");
})
```

2. executeSql()方法

executeSql()方法通常用来执行 SQL 语句，其调用格式如下所示：

```
executeSql(sqlString,[Arguments],SuccessCallback,ErrorCallback);
```

其中，参数 sqlString 表示需要执行的 SQL 语句；参数 Arguments 表示语句需要的实参；参数 SuccessCallback 表示 SQL 语句执行成功时的回调函数；参数 ErrorCallback)表示 SQL 语句执行出错时的回调函数。

例如，executeSql()方法的正确使用方法如下：

```
executeSql("insert into student value(?,?,?,?)",[01,"王倩",22,556]);
```

其中，“?”形参的数量必须与后面的实参数量完全对应，如果 SQL 语句中没有“?”形参，第二个参数中不允许用户有任何内容出错，否则执行 SQL 语句时将会报错。

9.4.3 实战——实现基于数据库的收藏夹管理

在 9.3 节我们详细介绍了客户端的数据存储机制，并使用 JSON 方式制作了一个收藏夹的案例。在学习了前面 HTML 5 本地数据库的知识之后，可以实现一个带数据库的收藏管理，包括查看收藏夹列表，以及收藏网站的添加、修改和删除。具体步骤如下。

步骤 01 新建一个 HTML 页面。在页面中制作收藏夹的添加表单和结果显示表格，代码如下：

```
<form onload="showDialyList()">
```

```
<div id="content">
    <h2>管理收藏夹</h2>
    网站编号：<input name="sitename" type="text" id="siteid" />(不能重复)
    <br/>
    网站名称：<input name="sitename" type="text" id="sitename" />   <br/>
    网站地址：<input name="siteurl" type="text" id="siteurl" />   <br/>
    网站备注：<input name="siteurl" type="text" id="sitetext" />   <br/>
    <input name="按钮" type="button" value="保存"
      onclick="btnAddDialy()"/>
    <input type="button" id="btnNew" onclick="editDialy('editid')"
      value="更改" />  <hr/>
    <p id="pstate"></p>
    <div class="text">
      <table width="100%" cellspacing="1" cellpadding="1" border="0"
        class="mytable">
        <tr>
          <th>编号</th>
          <th>网站名称</th>
          <th>网站 URL 地址</th><th>备注</th>
          <th>操作详情</th>
        </tr>
        <tbody id="list"></tbody>
      </table>
    </div>
  </div>
</body>
```

步骤 02 上段代码中，页面加载时调用函数 showDialyList()。该函数实现加载全部日志列表的功能。具体代码如下所示：

```
function showDialyList()
{
    db = openDatabase('DialyManage','2.0','收藏夹管理系统',10*1024*1024);
    if(db)
    {
        db.transaction(function(tx){
            tx.executeSql('select * from siteinfo',[],
                function(tx,rs){
                    var strHTML = "";
                    if(rs.rows.length>0)
                    {
                        for(var i=0;i<rs.rows.length;i++)
                        {
                            var intid = rs.rows.item(i).id;
                            strHTML += "<tr>"
                            strHTML += "<td>"+intid+"</td>";
                            strHTML +=
                              "<td>"+rs.rows.item(i).name+"</td>";
                            strHTML +="<td>"+rs.rows.item(i).url+"</td>";
                            strHTML +=
                              "<td>"+rs.rows.item(i).memo+"</td>";
                            strHTML +=
                              "<td><a href='javascript:showeditDialy("
                              +intid+")'>编辑</a>   ";
```

```
                        strHTML += "<a href='javascript:deleteDialy("
                          +intid+")'>删除</a></td>";
                        strHTML += "</tr>";
                    }
                }else
                {
                    strHTML +=
            "<tr><td colspan=5><center>暂时没有记录。</center></td></tr>";
                }
                  $("list").innerHTML = strHTML;
              },
              function(tx,ex){ $("pstate").innerHTML =
                  "显示列表失败,失败原因是: "+ex.message;
              }
          )
        });
    }
}
```

这段代码中，首先调用 openDatabase()方法创建或者打开了一个数据库名称是“DialyManage”、版本为 2.0 的 10MB 的数据库。然后使用 transaction()方法执行事务，调用 executeSql()方法执行查询，执行的 SQL 语句为“select * from siteInfo”。执行成功后，则可以访问成功回调函数中的 rs.rows.length 来获得查询记录的总条数，根据“rs.rows.item(Index 索引值).字段名”获得某个字段的值。同时在页面增加了“编辑”链接和“删除”链接，可以实现编辑数据和删除数据的功能。

步骤 03 单击“保存”按钮触发按钮的 click 事件，调用 btnAddDialy()函数实现添加收藏夹的功能。具体代码如下：

```
function btnAddDialy()
{
    var addid = $('siteid').value;
    var addname= $('sitename').value;
    var addurl = $('siteurl').value;
    var addtext= $('sitetext').value;
    db = openDatabase('DialyManage','2.0','收藏夹管理系统',10*1024*1024);
    if(db)
    {
        var addsql = "insert into siteinfo values(?,?,?,?)";
        db.transaction(function(tx){
            tx.executeSql('create table if not exists siteinfo(
              id unique,name text,url text,memo text)');
            tx.executeSql(addsql,[addid,addname,addurl,addtext],
                function(){ $("pstate").innerHTML =
                  "添加一条记录成功";showDialyList();},
                function(tx,ex){ $("pstate").innerHTML =
                  "添加一条数据记录失败,失败原因是: "+ex.message;}
            )
        });
    }
}
```

在 btnAddDialy()函数中有两条 SQL 语句，第一条 SQL 语句的功能是：如果名为

siteInfo 的表不存在，则创建 siteInfo 表。该表包含 4 个字段：id、name、url 和 memo。其中，字段 id 为主键，不允许重复；其他 3 个字段为字符型，它们分别表示网站名称、网站地址和备注。第二条 SQL 语句的功能是：实现向 siteInfo 表中添加记录的功能。执行成功后，则可以访问成功的回调函数，重新调用 showDialyList()函数加载数据记录。

步骤 04 单击某条记录的“编辑”链接，实现更改数据记录的功能。JavaScript 中的具体代码如下所示：

```
function showeditDialy(id)
{
    db = openDatabase('DialyManage','2.0','日志管理系统',10*1024*1024);
    if(db)
    {
        db.transaction(function(tx){
            tx.executeSql("select * from siteInfo where id='"+id+"'",[],
            function(tx,rs){
                $('siteid').value = rs.rows.item(0).id;
                //省略日志输入框赋值
            },
            function(tx,ex){ $("pstate").innerHTML =
               "获取数据失败: "+ex.message;}
            )
        });
    }
}
function editDialy(editid)
{
    db = openDatabase('DialyManage','2.0','收藏夹管理系统',10*1024*1024);
    var editid = $('siteid').value;
    var editname = $('sitename').value;
    var editurl = $('siteurl').value;
    var edittext=$('sitetext').value;
    var editsql = "update siteInfo set name=?,url=?,memo=? where id=?";
    if(db)
    {
        db.transaction(function(tx){
            tx.executeSql(editsql,[editname,editurl,edittext,editid],
                function(){ showDialyList(); },
                function(tx,ex){ $("pstate").innerHTML =
                  "更新数据记录失败,失败原因是: "+ex.message;}
            )
        });
    }
}
```

showeditDialy()函数首先根据收藏夹的 id 获得网站的详细信息，执行的 SQL 语句为“select * from siteInfo where id = 传入的 id”，这条 SQL 代码中没有使用“?”参数，而是直接将参数加到 SQL 语句后面。页面信息修改完成后，单击“更改”按钮提交用户修改记录，执行的 SQL 语句为“update siteInfo set name=?,url=?,memo=? where id=?”。执行成功后，可以访问成功的回调函数，调用函数 showDialyList()重新加载数据记录。

步骤 05 收藏夹查看、添加和编辑功能实现以后，我们还需要实现删除收藏夹的功

能。单击某条记录的“删除”链接删除某条收藏夹的功能。JavaScript 中的具体代码如下所示：

```
function deleteDialy(delid)
{
    db = openDatabase('DialyManage','2.0','收藏夹管理系统',10*1024*1024);
    var delsql = "delete from siteInfo";
    if(delid!=0)
    {
        delsql = "delete from siteInfo where id='"+delid+"'";
    }
    if(db)
    {
        db.transaction(function(tx){
            tx.executeSql(delsql,[],
                function(){ showDialyList();},
                function(tx,ex){ $("pstate").innerHTML =
                  "删除数据记录失败,失败原因是: "+ex.message;}
            )
        });
    }
}
```

上段代码中根据传入的 delid 参数判断是全部清空还是删除某条记录。执行 SQL 语句删除成功后可以访问回调函数中的代码，最后调用 showDialyList()函数加载记录。

步骤 06 目前为止，基于 HTML 5 本地数据库实现收藏夹管理的增、删、改、查功能已经完成。上面代码的运行效果如图 9-17 所示。



图 9-17　收藏夹管理

9.5　跨文档传输信息

为了代码的安全性，在 JavaScript 脚本中不允许跨域访问其他页面中的元素，这给不同页面数据的互访带来了障碍。HTML 5 解决了这个问题，允许在两个不同的域名与端口之间接收发送数据的功能。

首先，需要接收从其他的窗口发送过来的消息，这就必须对窗口对象的 message 事件进行监视。监视代码如下：

```
window.addEventListener("message", function(){}, false);
```

postMessage()方法是 HTML 为了解决跨域通信特别引入的一个新的解决办法，目前许多主流的浏览器都支持这一方法。postMessage()方法允许页面中的多个 iframe/window 的通信，postMessage()方法也可以实现 Ajax 直接跨域，而不通过服务器端代理。

postMessage()方法向其他窗口发送消息，定义如下所示：

```
otherwindow.postMessage(message, targetOrigin);
```

上述语法中，otherwindow 表示接受数据页面的引用对象，可以通过 window.open()方法返回该对象，也可以通过下标返回 window.frames 的单个实体对象，还可以是 iframe 的 contentWindow 属性。另外，postMessage()方法包含两个参数，具体说明如下。

- message：表示所发送的消息文本，可以是 JSON 对象转换后的字符内容。
- targetOrigin：表示发送数据的 URL 来源，可以在 URL 地址字符串中使用通配符“*”指定全部地址。

【例 9.9】

创建一个示例，在主页面输入一个数，将该数传递到另一个页面进行处理，再到主页面中显示处理结果。具体步骤如下。

步骤 01　创建一个 HTML 页面，并添加如下简单代码：

```
请输入要删除的文章编号：
<input type="text" id="strid" />
<input type="button" onclick="test()" value="确定" />
<br/><p id="result"></p>
<iframe id="frame11" src="calc.html" style="width:0px; height:0px;">
</iframe>
```

在上述代码中创建了 iframe 框架，并且指向的一个 HTML 页面 calc.html，该页面用于完成具体的计算并返回结果。id 为 result 的 p 元素用于显示结果。

步骤 02　接下来通过 JavaScript 代码实现将值传递给子页面，子页面返回处理后的信息，传给本页面。JavaScript 代码如下所示：

```
<script type="text/javascript">
var strOrigin = "http://localhost";
window.onload = function () {
    window.addEventListener("message", function (event) {
        if (event.origin == strOrigin) {
            $("result").innerHTML = event.data;
        }
        else {  return; }
        }
    , false);
}
function test() {
    var str =$("strid").value;
     $("frame11").contentWindow.postMessage(str, strOrigin);
```

```
}
function $(id){return document.getElementById(id);}
</script>
```

步骤 03 上述 JavaScript 代码完成了获取值，并且将值传递给子页面，在子页面中将处理接收的值并且返回处理后的信息。子页面 calc.html 的代码如下所示：

```
<script type="text/javascript">
var strOrigin = "http://localhost";
window.onload = function () {
    window.addEventListener("message", function (event) {
        if (event.origin == strOrigin) {
            var strhtml = "";
            var id = event.data;
            if (id > 0 && id <= 100) {
                strhtml = "编号为"+id+"的文章已删除。";
            }
            else if (event.data > 100) {
                strhtml = "找到编号为"+id+"的文章。";
            }
            else {
                strhtml = "请输入正确格式的编号。";
            }
           event.source.postMessage(strhtml, strOrigin);
        }
    }
    , false);
}
</script>
```

步骤 04 上述代码的运行效果如图 9-18 所示。



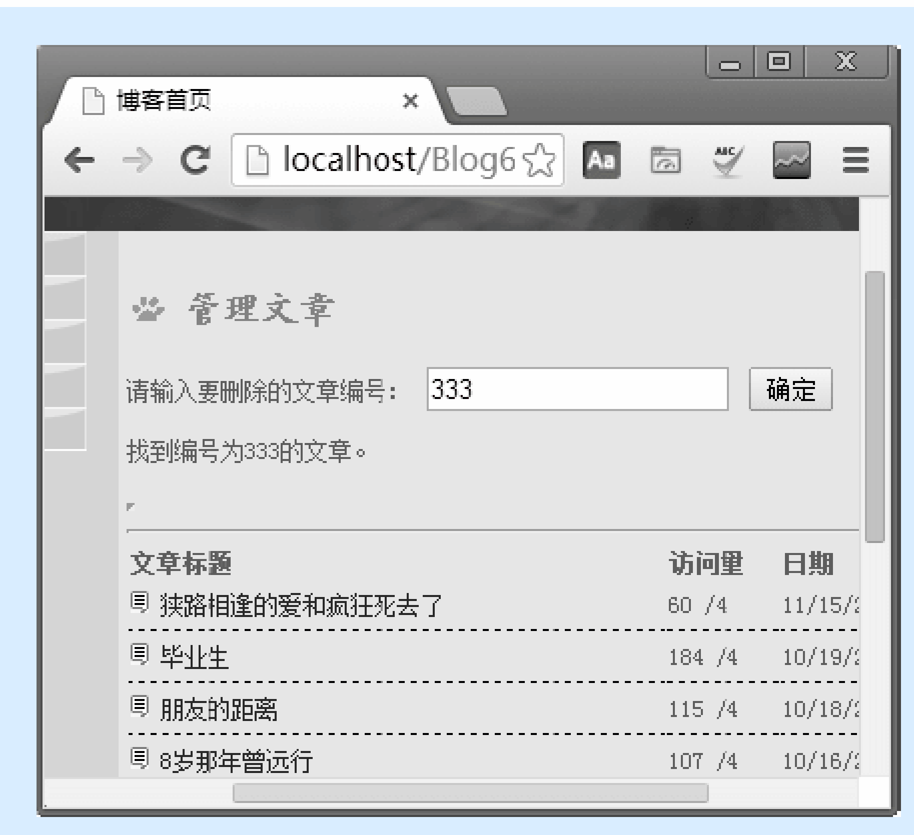


图 9-18 跨域传输信息的运行效果

9.6 多 线 程

在 HTML 5 之前，浏览器中的 JavaScript 脚本代码都是以单线程方式运行的，虽然有

多种方式实现了对多线程的模拟(如 JavaScript 中的内置函数 setInterval()和 setTimeout()等)，但是，本质上，程序的运行仍然是由 JavaScript 引擎以单线程调度的方式进行的。

HTML 5 中引入的多线程使得浏览器端的 JavaScript 引擎可以并发地执行 JavaScript 代码，从而实现了对浏览器端多线程编程的良好支持。下面将详细介绍如何使用 HTML 5 中的多线程。

9.6.1 认识HTML 5 多线程

多线程是 HTML 5 中非常重要的一个功能，它解决先前版本中脚本耗费时间长、中断执行的处理及长时间无反应的情况。

HTML 5 为多线程提供了 Worker 对象。通过调用 Worker 对象，可以将前台 JavaScript 代码分割成若干代码块，分别交给不同的后台线程处理，避免单线程执行缓慢的问题。

使用 HTML 5 多线程时，必须建立一个 Worker 对象，并且传入要被调用的 JavaScript 文件的地址，而且必须把 Worker 要执行的操作写入到这个 JavaScript 文件中。

1. 创建Worker对象

创建 Worker 对象很简单，只需要把在线程中执行的 JavaScript 文件的文件名传递给构造函数即可。

例如，下面的代码创建了一个 Worker 对象，并且传入 worker.js 文件：

```
var worker = new Worker("worker.js");
```

2. 线程通信

在主线程与工作线程间进行通信时，使用的是线程对象的 postMessage()方法和 onmessage 事件，不管是谁向谁发送数据，发送方使用的都是 postMessage()方法，接收方使用的都是 onmessage 事件。postMessage()方法有一个参数，即传递的参数；onmessage 事件也只有一个参数，即通过 event.data 获取收到的数据。

例如，onmessage 事件的简单代码如下：

```
worker.onmessage = function (e){
    alert(e.data);
}
```

3. 发送JSON数据

JSON 是 JavaScript 原生支持的内容，比较复杂的数据就可以使用 JSON 进行传递。如下面的代码：

```
postMessage({'cmd': 'init', 'timestamp': Date.now()});
```

4. 处理错误

当线程发生错误的时候，它的 onerror 事件回调函数会被调用，所以处理错误的方式很简单，就是挂接线程实例的 onerror 事件。这个回调函数有一个参数对象 error，这个对

象有 3 个常用的属性。

- message：错误消息。
- filename：发生错误的脚本文件。
- lineno：发生错误的行。

例如，onerror 事件的代码如下：

```
worker.onerror = function(e){
    alert("在第" + e.lineno + "行发生错误：" + e.message);
}
```

5. 销毁线程

线程内容可以调用 close()方法销毁自己的线程，在线程外部的主线程中，使用线程实例的 terminate()方法销毁线程。

9.6.2 实战——Worker对象的简单应用

上节已经简单介绍了如何创建和使用 Worker 对象，下面主要通过一个案例来演示如何使用 Worker 对象处理线程。

本案例中，页面加载时创建一个 Worker 后台线程，当输入内容完成后，单击按钮向后台线程发送输入内容，后台线程将数据处理完成后，返回前台调用并输入消息。实现该功能的具体步骤如下。

步骤 01 添加新的 HTML 页面，在页面的合适位置添加 textarea 元素、input 元素和 div 元素等。其具体代码如下：

```
内   容：
<label><textarea name="textarea" id="textarea" cols="0" rows="0"
class="detail content" placeholder="请输入您的评价内容"></textarea></label>
评论人：<label><input type="text" name="cname" id="commentname" cols="0"
rows="0" /></label>
<div><a href="javascript:;" onClick="btnSend();" class="button">提 交
</a></div>
<div id="showSapn"></div>
```

步骤 02 在 JavaScript 中首先判断浏览器是否支持 Worker 对象，如果支持，则直接创建该对象，如果不支持则弹出提示。具体代码如下：

```
var objWorker;
if(typeof(Worker)!=="undefined")
{
    objWorker = new Worker("comments.js");
}
else
{
    alert("该浏览器不支持 Web Worker");
}
```

步骤 03 单击“提交”按钮时触发该按钮的 onclick 事件，调用 btnSend()函数，该

函数的具体代码如下：

```
function btnSend()
{
    var content = document.getElementById("textarea").value;   //评论内容
    var name = document.getElementById("commentname").value;   //评论人
    var con = name + "@" + content;                            //组合结果
    objWorker.postMessage(con);                                //发送信息
}
```

上述代码中，首先获取用户输入的评论内容和评论人，然后将它们组合保存到变量 con 中，最后调用 objWorker 对象的 postMessage()方法发送内容。

步骤 04 创建名称为 comments.js 的文件，在该文件中接收页面传过来的内容。该文件中的具体代码如下：

```
onmessage = function(event){
    var datas = event.data;                         //获取从页面中传入的数据
    var splits = datas.split('@');                  //以@进行拆分
    var result = "";                                //声明变量保存最后的信息
    result = "[" + splits[0] + "]的评论内容是：" + splits[1];
    postMessage(result);
}
```

上述代码首先调用 event 对象的 data 对象获取从页面中传入的数据，然后调用 split()方法拆分以@为主的内容，最后调用 postMessage()方法将保存到 result 变量中的内容返回到上个页面。

步骤 05 调用 postMessage()方法时，会触发 onmessage 事件，在页面中通过 objWorker 对象的该事件来获取信息并接收，最后将内容显示到 div 元素中。其具体代码如下：

```
objWorker.onmessage = function(event)
{
    document.getElementById("showSapn").innerHTML = event.data;
}
```

步骤 06 运行本案例，输入内容后单击“提交”按钮，查看效果，最终运行效果如图 9-19 所示。



图 9-19　多线程的运行效果

9.7 获取位置信息

HTML 5 中新增加了 Geolocation API 接口获取地理位置，它允许用户在 Web 应用程序中共享他们的位置，使其能够享受位置感知服务，包括了解周围情况等。地理位置信息的来源有经度、纬度和其他特性，获取这些数据的途径有 GPS、WiFi 和蜂窝站点等。

下面详细介绍 HTML 5 使用 Geolocation API 接口获取地理位置的方法。

9.7.1 认识地图API

HTML 5 为 window.navigator 对象增加了一个 geolocation 属性，该属性返回一个 Geolocation 对象。调用 Geolocation 对象提供的方法，即可实现获取位置信息。

因此，在使用之前，可以检测浏览器是否支持 Geolocation 对象。代码如下：

```
function checkDemo(){
    if(navigator.geolocation){
        alert("支持 HTML 5 Geolocation 对象。");
    }else{
        alert("不支持 HTML 5 Geolocation 对象。"));
    }
}
```

Geolocation 对象常用的方法有 getCurrentPosition()方法、watchCurrentPosition()方法和 clearWatch()方法。

1. getCurrentPosition()方法

getCurrentPosition()方法可以获取用户当前的地理位置信息，该方法的语法形式如下：

```
window.navigator.geolocation.getCurrentPosition(onSuccessCallback,
  onErrorCallback,options);
```

上述语法中 getCurrentPosition()方法传入了 3 个参数，第一个参数用于成功获取当前地理位置时的回调函数，该函数中需要传入形参对象 position，该对象在下一节进行介绍；第二个参数用于获取当前地理位置失败时的回调函数；第三个参数是一个可选择的对象，表示一些属性内容。

2. watchCurrentPosition()方法

watchCurrentPosition()方法用于持续获取用户的当前地理位置信息，它会定期地自动获取。该方法的语法形式如下：

```
int watchCurrentPosition(onSuccessCallback,onErrorCallback,options);
```

上述语法中为 watchCurrentPosition()方法传递了 3 个参数，这 3 个参数的使用方法与 getCurrentPosition()方法中的参数相同。该方法返回值是一个数值，该数值可以被 clearWatch()方法使用，表示停止对当前地理位置信息的监视。

3. clearWath()方法

clearWath()可以停止对当前用户的地理位置信息的监视。其语法形式如下：

```
void clearWath(watchId);
```

使用该方法时需要向该方法中传递形参，它的值为调用 watchCurrentPosition()方法监视地理位置信息时的返回参数。

9.7.2 Position对象

无论是 Geolocation 对象的 getCurrentPosition()方法还是 watchCurrentPosition()方法，都需要传入一个调用成功后的回调函数作为参数。该回调函数可以使用 Position 对象有关的属性来显示当前的位置信息。

Position 对象包含两个重要属性：timestamp 和 coords。timestamp 属性表示获取地理位置时的时间，而 coords 属性则包含多个属性值，具体说明如表 9-7 所示。

表 9-7　coords 属性所包含的值

值 名 称	说　明
accuracy	当前地理位置的精确度
latitude	当前地理位置的纬度
longitude	当前地理位置的经度
altitude	当前地理位置的海拔高度
altitudeAccuracy	当前地理位置的海拔精确度(单位：米)
heading	当前设置的前进方向，用面朝正北方向的顺时针旋转角度来表示。无法获取时返回值为 null
speed	当前设置的前进速度，以米/秒为单位，无法获取时返回值为 null

【例 9.10】

下面创建一个示例，演示如何通过 position 对象的相关属性获取用户当前地理位置信息。实现该功能的步骤如下。

步骤 01 添加新的 HTML 页面，在页面的合适位置添加 span 元素，该元素显示详细的地理信息。页面相关的具体代码如下：

```
<span id="ShowMessage"></span>
```

步骤 02 页面加载时调用 init()函数自动获取信息，该函数的具体代码如下：

```
function init()
{
    if(navigator.geolocation)                          //判断浏览器是否支持
    {
        navigator.geolocation.getCurrentPosition(
            handle getInfo,                            //成功时调用的函数
            handle_error,                              //失败时调用的函数
```

```
            {                                       //其他属性信息的设置
                maximumAge:5*1000*60,                   //缓存有效时间
                timeout:5000                            //超时时间限制
            }
        )
    }else{
        alert("浏览器不支持当前位置的显示功能");
    }
}
window.addEventListener("load",init,true);      //页面加载时调用 init()事件
```

上述代码中，首先判断浏览器是否支持显示当前位置的功能，如果支持成功时，调用 handle_getInfo()函数，失败时，调用 handle_error()函数且设置其他属性的相关信息。

步骤 03 handle_getInfo()函数成功时才执行，在该函数中传递一个 position 对象参数，然后调用该对象的相关属性显示详细内容。其具体代码如下：

```
function handle_getInfo(position)
{
    var strHTML = "";
    var objInfo = position.coords;
    strHTML += "当前位置的纬度值：<b>" + objInfo.latitude + "</b><br/>";
    strHTML += "当前位置的经度值：<b>" + objInfo.longitude + "</b><br/>";
    strHTML += "当前位置的精确度：<b>" + objInfo.accuracy + "</b><br/>";
    strHTML += "当前位置的前进速度：<b>" + objInfo.speed + "</b><br/>";
    strHTML += "当前位置的前进方向：<b>" + objInfo.heading + "</b><br/>";
    strHTML += "当前位置的时间戳：<b>" + objInfo.timestamp + "</b><br/>";
    document.getElementById("ShowMessage").innerHTML = strHTML;
}
```

上述代码中首先声明全局变量 strHTML，保存要显示的内容，接着调用 coords 属性的多个属性值分别显示经度值、纬度值、精确度及前进速度和前进方向等，然后直接调用 position 对象的 timestamp 属性显示时间戳，最后将变量的内容显示到 id 为 ShowMessage 的 span 元素中。

步骤 04 handle_error()函数失败时才会执行，在该函数中传递一个 error 对象参数，然后根据 code 属性的值判断显示不同的内容。其具体代码如下：

```
function handle_error(error)
{
    switch(error.code){                             //获取 code 属性的值
        case 0:
            alert("出现了未知的错误！");
            break;
        case 1:
            alert("位置服务被拒绝");
            break;
        case 2:
            alert("暂时获取不到位置信息");
            break;
        case 3:
            alert("获取信息超时");
            break;
    }
```

```
}
```

步骤 05　运行本案例的代码进行测试，页面的最终效果如图 9-20 所示。



图 9-20　显示地理位置的详细内容

9.8　HTML 5 的离线缓存特性

随着 Web 应用的普及，Web 离线应用显得尤为重要。因为 Web 应用程序要时刻与服务器保持交互才能正常工作。一旦中断 Web，相关应用也随之停止。在 HTML 5 中提供了一个供本地缓存使用的 API，使用这个 API 可以实现离线 Web 应用程序的开发。

离线 Web 应用程序是指当客户端与 Web 应用程序的服务器没有建立连接时，也能够正常地在客户端本地使用该 Web 应用程序进行有关的操作。HTML 5 中引用了离线应用程序缓存，使得在无网络连接状态下运行应用程序成为可能，有了它 Web 应用程序，就可以在没有网络连接的情况下运行。而且还可以指定 HTML 5 应用程序中具体哪些资源(如 CSS、JavaScript 和 HTML 等)脱机时可用。

使用离线存储避免了加载应用程序时所需的常规网络请求，如果缓存清单文件是最新的，浏览器就不用检查其他资源是否为最新了。大部分应用程序可以从本地应用缓存中加载完成，另外从缓存中加载资源可节省带宽，这对于移除 Web 应用是相当重要的。

既然使用本地缓存的 API 可以实现离线 Web 应用程序的开发，那么它和浏览器网页缓存存在哪些区别呢？如下几点列出了比较明显的区别。

(1)　本地缓存是为了整个 Web 应用程序服务的；而浏览器的网页缓存是只服务于单个网页的。

(2)　本地缓存只缓存那些指定需要缓存的网页；任何网页都具有网页缓存。

(3)　本地缓存是可靠的，开发人员可以控制对哪些内容进行缓存，哪些内容不进行缓存；网页缓存是不安全、不可靠的。

(4)　开发人员可以通过编程的方法来控制缓存的更新，利用缓存对象的各种属性、状态和事件等开发强大的离线应用程序。

创建一个 HTML 5 的离线缓存应用是非常简单的，以下是快速创建离线应用程序缓存的步骤。

步骤 01　创建一个有效的 HTML 5 文件。

HTML 5 中直接使用 DOCType 创建 index.html 页面文件比 HTML 4 中使用 XHTML 创建文件时更加易于识记。

步骤 02 新增加对.htaccess 的支持。

一个 manifest 文件需要被正确的 MIME-TYPE 支持，这种文件类型为“text/cache-manifest”，必须在服务器中进行配置。假设所使用的服务器是 Apache，那么在创建文件之前，需要向.htaccess 文件中新增加一个指令。

打开该.htaccess 文件，该文件部署在网站的根目录下，新增以下代码：

```
AddType text/cache-manifest .manifest
```

步骤 03 创建.manifest 文件。

步骤 04 将 manifest 文件链接到 HTML 文档中。

为了启动应用缓存，因此需要在网页的<html></html>标记中添加 manifest 属性，将该属性的值指定为.manifest 文件。

步骤 05 在多个浏览器中进行测试。

注意：将网页设置 manifest 属性后，当用户访问每一个包含该属性的页面时，都会被缓存。如果 manifest 属性没有被指定，则不会缓存(除非网页被直接在 manifest 文件中指定)。

【例 9.11】

本节练习非常简单，在网页中显示 3 张不同的图片，并且在缓存清单中离线缓存两张图片，步骤如下所示。

步骤 01 创建一个新的页面，在页面中添加 3 张图片，其中前两张图片来自网络，最后一张图片来自本项目的 image 文件夹。代码如下：

```
<img src="http://www.baidu.com/img/bdlogo.gif" title="百度" height= "60"
/> <img src="http://www.itzcn.com/images/logo.gif" title="窗内网"
height="60"/> <img src="images/google.png" title="谷歌" height="60" />
```

步骤 02 创建全称是 base.manifest 的缓存清单文件，并且在该文件中定义要缓存的内容。基本文件内容如下：

```
CACHE MANIFEST
#version 5.2.0

CACHE:
http://www.baidu.com/img/bdlogo.gif
images/google.png
```

步骤 03 将上一步中创建的 base.manifest 文件链接到 HTML 网页中，在网页中的<html></html>标记中添加 manifest 属性，将此属性指定为 base.manifest 文件：

```
<html manifest="base.manifest">
```

步骤 04 在 IIS 中访问页面进行测试，大多数浏览器都不会提醒是否允许缓存，而是默认自动缓存。如图 9-21 所示为 Chrome 浏览器的效果。

步骤 05 以手动的方式断开网络连接，然后重新访问该页面查看效果，离线效果如图 9-22 所示。



图 9-21 Chrome浏览器连接网络访问图片



图 9-22 Chrome浏览器离线访问缓存的图片

9.9 本 章 习 题

1. 填空题

(1) 使用 file 对象的__________属性可以获取不带路径的文件名称。

(2) 使用__________代码可以判断当前浏览器是否支持 FileReader 接口。

(3) 在 FileReader 接口中__________方法负责读取文本文件。

(4) 使用 sessionStorage 对象读取数据需要使用__________方法。

(5) 当用户关闭浏览器窗口后，数据不会被保存指的是__________对象。

(6) dataTransfer 对象的__________方法可以为元素添加指定的数据。

(7) HTML 5 中可以在页面调用__________方法解决跨域通信的问题。

(8) 调用 Geolocation 对象的__________方法能够实时地获取或者检测用户的地理位置信息。

(9) __________文件也叫清单文件，它以清单的形式列举了需要被缓存或不需要被缓存的资源文件的名称。

2. 选择题

(1) 下列关于 FileReader 接口中事件的描述，不正确的是__________。

A. 总是先执行 onloadstart 事件，再执行 onloadend 事件

B. 如果执行了 onerror 事件，将不会执行 onloadend 事件

C. 只有在 onload 事件中能获取内容

D. 在 onload 事件和 onloadend 事件中都可以获取内容

(2) 调用 abort()方法将触发 FileReader 接口的__________事件。

A. abort　　B. onabort　　C. onerror　　D. onend

(3) 拖放元素将源文件拖放到目标文件上时触发的事件是__________。

A. dragend　　B. dragover　　C. dragenter　　D. drop

(4) HTML 5 中接受服务器端发送来的数据时触发的事件是__________。

A. onopne B. onerror 事件 C. onmessage 事件 D. mark 元素

(5) 后台线程调用 postMessage()方法发送数据后，前台页面会触发 message 事件，并且在该事件中获取处理后的数据。下段代码空白处应该填写__________。

```
worker.addEventListener("message",function(event){
    var content =___________;  //后台处理完成后返回到前台的数据
},false)
```

A. e.message B. event.message C. event.data D. e.data

(6) 如果想使用 sessionStorage 对象写入键名为 name，值为“陈虎”的数据，下面__________的写法是正确的。

A. sessionStorage.setItem(“name”, “陈虎”)

B. localStorage.setItem(“name”, “陈虎”)

C. sessionStorage.getItem(“陈虎”)

D. sessionStorage.setItem(“陈虎”, “name”)

(7) __________方法不属于 navigator.geolocation 对象。

A. clearWatch() B. getCurrentPosition()

C. watchCurrentPosition() D. postMessage()

(8) Position 对象的 coords 属性的值有多个，通过______________可以获取当前位置的经度。

A. accuracy B. latitude C. longitude D. altitude

3. 上机练习

(1) 处理文件读取时的错误

根据本章 9.1 节学习的知识，制作一个读取文件内容的案例。要求在读取时会对读取事件进行监听，并处理读取异常时的错误，最后显示到页面上。

如图 9-23 所示为先选择一个.txt 文件，再修改该文件的名称，然后单击“上传”按钮之后的错误处理效果。



图 9-23 错误处理效果

(2) 日志管理

根据本章 9.4 节学习的知识制作一个基于 HTML 5 数据库的日期管理功能。要求具有

添加日志、编辑日志、删除日志和清空日志的功能。示例的参考运行效果如图 9-24 所示。



图 9-24　日志管理效果

(3)　多线程计算器

根据本章 9.6 节学习的知识，制作一个多线程的计算器。要求用户可以在页面输入要计算的两个操作数，然后选择要执行的操作，最后显示计算结果，同时可以进行停止以及清空等操作。示例的参考运行效果如图 9-25 所示。



图 9-25　多线程计算器的效果





第 10 章

CSS 3 快速入门

CSS(Cascading Style Sheets，层叠样式表)在网页制作中经常会被使用到，它与 HTML 和 JavaScript 结合，能够制作出绚丽夺目的网页。使用 CSS 只要对相应的代码做出简单修改，就可以改变同一页面的不同部分，或者页数不同的网页的外观和格式。CSS 3 是 CSS 技术的升级版本，CSS 3 语言开发是朝着模块化发展的。与先前的版本相比，CSS 3 增加了许多新的功能，也在原来的基础上对某些内容做出了修改。

本章向读者介绍 CSS 3 的基础知识，通过本章的学习，读者不仅可以了解 CSS 3 的发展和优缺点，还可以掌握 CSS 3 中新增的颜色，也能够对 CSS 3 中的新增属性有一定的认识。

本章学习目标：

- 了解 CSS 3 的发展历史
- 了解 CSS 3 的优缺点
- 熟悉浏览器对 CSS 3 的支持情况
- 掌握 HSL 和 HSLA 的使用
- 掌握 RGBA 和 Opacity 的使用
- 熟悉 CSS 3 中新增加的选择器
- 了解 CSS 3 中新增的多种属性

10.1 了解 CSS 3

CSS 3 是 CSS 的升级版本，也是对 CSS 2 的扩展，它增加和修改了许多新的功能。下面简单了解 CSS 3 的基础知识，包括它的发展、优缺点和浏览器的支持情况。

10.1.1 CSS 3 发展概述

在 20 世纪 90 年代初，HTML 语言诞生，各种形式的样式表也开始出现。各种不同的浏览器结合自身的显示特性，开发了不同的样式语言，以便于读者自己调整网页的显示效果，这时的样式语言仅供读者使用，而不是供设计者使用。随着 HTML 的发展，1994 年初，哈坤 • 利提出了 CSS 的最初想法，伯特 • 波斯当时正在设计 Argo 浏览器，他们决定共同开发 CSS。

1994 年年底，哈坤在芝加哥的一次会议上第一次展示了他们对 CSS 的构想。在 1995 年底，CSS 语言正式完成，同年 12 月发布了第一个版本。1998 年 5 月，CSS 2 正式发布。之后，W3C 完成了 CSS 3 的工作草案，在 2001 年 5 月 23 日发布 CSS 3，这是目前 CSS 的最新版本。发展历程如下：

- 2002 年 5 月 15 日发布 CSS 3 中规范文本行模型的 line 模块；同年，还发布了具有列表样式和新增边框功能的 Lists 模块和 Border 模块。
- 2003 年依次发布了定义 CSS 3 生成及更换内容的 Generated and Replaced Content 模块，同年，还发布 Syntax 模块和 Hyperlink Presentation 模块，它们分别表示演示效果与新定义的 CSS 语法规则。
- 2004 年 2 月发布 CSS 3 Hyperlink Presentation 模块，该模块定义了超链接表示规则；同年 12 月发布了 CSS 3 Speech 模块，该模块定义了语音样式规则。
- 2005 年 12 月 15 日发布 Cascading and Inheritance 模块，它重新定义了 CSS 层叠和继承规则。
- 2007 年先后发布 Basic Box 模块和 Grid Positioning 模块，它们分别用于定义 CSS 基本盒模型与 CSS 网格定位规则。
- 2009 年新定义了许多新的模块，包括 Animations(动画模型)、3D Transforms(3D 转换)、Fonts(字体)、Image Value(图像内容显示)、Flexible Box Layout(灵活的框布局)、视图、Transitions(动画过渡)以及 2D Transforms(2D 转换)等。
- 2010 年 4 月发布 Template Layout(模板布局)和 Generated Content For Page Media (分页媒体内容)两个模块；同年 10 月又发布了 Text(文本)和 Background and Borders(边框和背景)两处模块。

10.1.2 CSS 3 的优缺点

CSS 3 已经完成了多个模块的开发，它给设计者和开发者带来了全新的体验。CSS 3 将完全向后兼容，因此，没有必要修改现在的设计来让它们继续运作，网络浏览器也将继

续支持 CSS 2。CSS 3 主要的影响是可以使用新的可用的选择器和属性，这些选择器和属性允许开发者实现新的设计效果(例如渐变和动画)，而且还可以使用更加简单的方式实现已有的效果(例如背景和分栏)。

已经有学者预言“CSS 3 和 HTML 5 将改变未来的 Web 世界”。广大开发者已经通过实践证明了 CSS 3 和 HTML 5 的强大。虽然目前 CSS 3 和 HTML 5 还没有完全普及，浏览器的支持也在初级实验阶段，但是这正是需要读者积极去学习和实践的时候，只有这样才不会落伍，才不会被淘汰。

CSS 3 在为开发者带有全新体验的同时，还存在着一些负面影响。开发者在使用时应该扬长避短，这样才能更好地理解和学习 CSS 3。

(1) 落伍的 IE 浏览器

CSS 3 新增的许多功能在 IE 浏览器中无法看到效果，但是随着 IE 浏览器新版本的发布，它对 CSS 3 和 HTML 5 的支持功能也在增加。出于安全性考虑，应该尽量避免将这些新功能用于重要的设计中，同时也应该保证，当这些效果不起作用时有替代的解决方案。

(2) 验证问题和代码冗余

目前的 CSS 3 规范并不是最终版本，不同的浏览器都在使用各自的私有属性实现新功能，这可能会为 CSS 验证埋下隐患。同时，这还使 CSS 代码显得十分冗余。

(3) 反面效果

设计页面时并不是一定要使用 CSS 3 的新增属性，不恰当的使用可能会带来一些反面效果，阴影效果就是一个例子。

10.1.3 浏览器支持情况

CSS 3 为广大 Web 开发者带来了全新的体验，但是并非所有的浏览器都提供对它的支持。各个主流的浏览器都定义了各自的私有属性，以便用户体验 CSS 3 的新特性。浏览器这种定义自己的私有属性的方法可以避免不同的浏览器在解析相同属性时出现冲突，但是也为开发者带来了麻烦，因为 Web 开发者不仅需要使用更多的 CSS 样式代码，而且还非常容易导致同一个页面在不同的浏览器之间表现不一致。

提示：网页不需要在所有浏览器中看起来都严格一致，有时候在某个浏览器中使用私有属性实现特定的效果也是可行的。

不同浏览器的内核有所不同，这导致浏览器定义的私有属性也有不同。WebKit 类型的浏览器(例如 Chrome)的私有属性是以-webkit-为前缀的，Gecko 类型的浏览器(例如 Firefox)的私有属性是以-moz-为前缀的，Opera 浏览器的私有属性是以-o-为前缀的，而 IE 浏览器的私有属性是以-ms-为前缀的。

在 Windows 系统下，各主流浏览器对 CSS 3 各个模块的支持情况有所不同。Web 开发者可以通过多个工具进行测试，下面介绍常用的两个。

1. Can I use

Can I use 的功能非常强大，它不仅可以检测浏览器对 CSS 3 的支持情况，也能检测

HTML 5 和 SVG 等其他内容。Can I use 的网址是 http://www.caniuse.com，页面打开效果如图 10-1 所示。



图 10-1　Can I use网页

开发者可以单击图 10-1 中的链接查看浏览器的支持情况，也可以在网页的搜索区域输入属性或者元素等内容进行搜索。例如，图 10-2 显示了各个浏览器对 Border-radius 属性的支持情况。

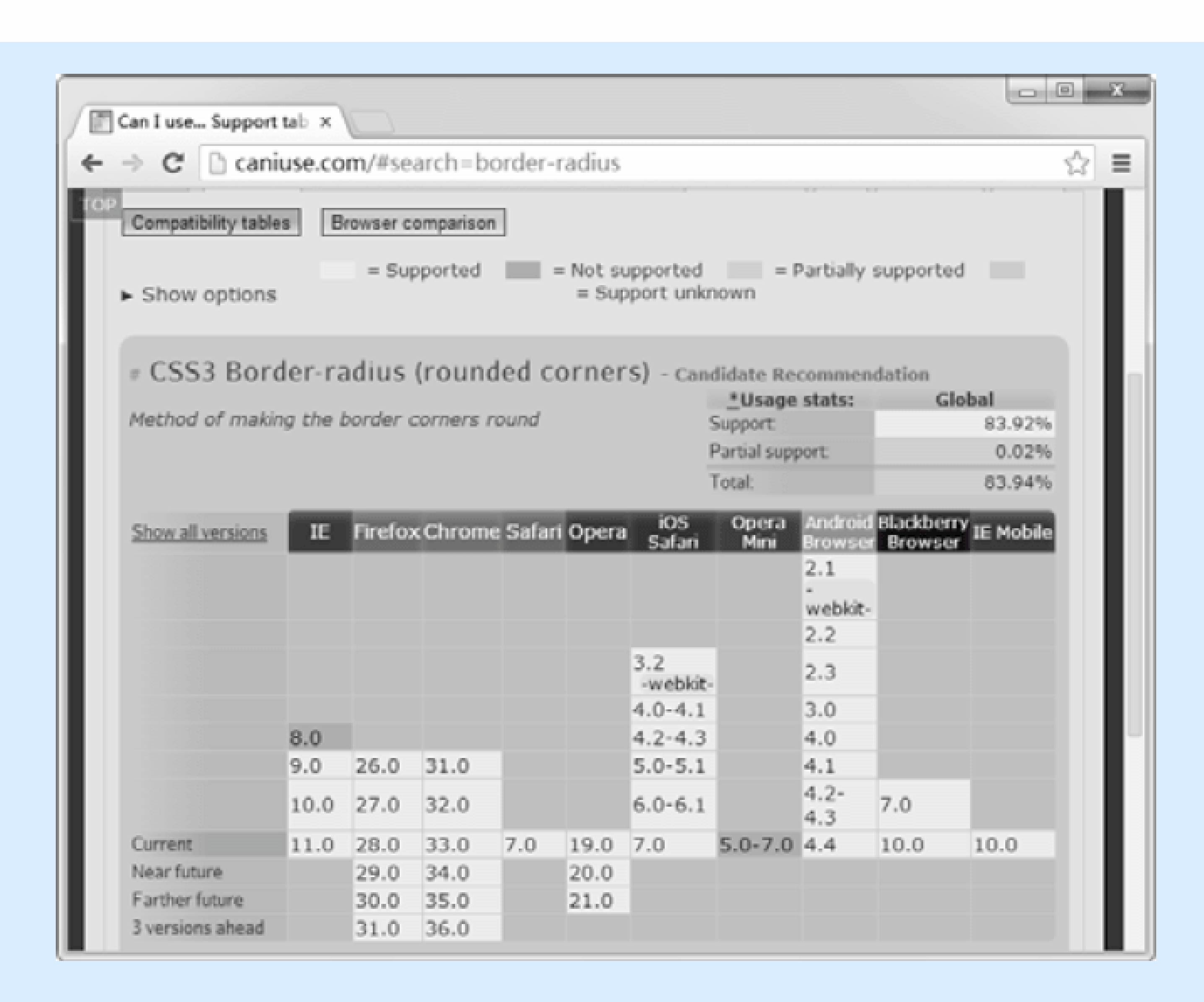


图 10-2　各个浏览器对Border-radius属性的支持情况

从图 10-2 中可以看出，该图显示了 IE、Firefox、Chrome、Safari 和 Opera 等多个浏览器对 Border-radius 属性的支持情况。其中，红色区域表示当前浏览器的版本不支持此属性，浅绿色区域表示对此属性支持，深绿色区域表示部分支持，灰色区域表示未知。另外，浏览器在使用时可能需要添加私有属性，例如-webkit-。

2. findmebyIP

findmebyIP 是一个简单实用的在线应用，帮助开发者检测使用的浏览器版本是否支持 CSS 3 和 HTML 5，支持哪些属性。findmebyIP 的网址是 http://fmbip.com/litmus/，打开效果如图 10-3 所示。图 10-3 中，针对 MAC 和 Windows 两个操作系统的 Chrome、Firefox、Opera、Safari 和 IE 浏览器进行了比较。

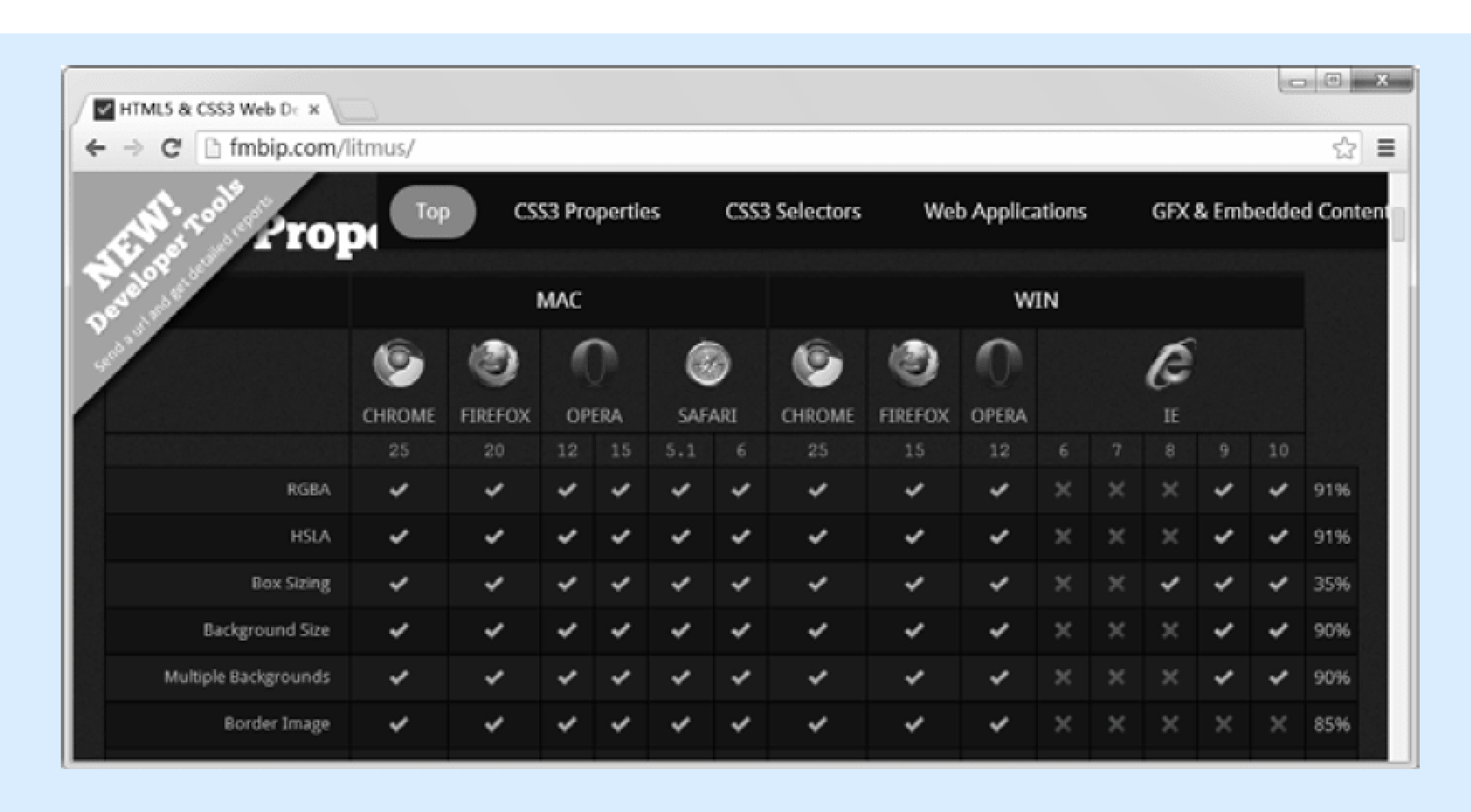


图 10-3　findmebyIP的网页

10.2　CSS 3 的新增颜色

CSS 3 相比先前的 CSS 2 有许多不同，模块化细分的同时也新增了功能。例如，开发者可以使用 RGB、十六进制或者英文颜色名称(例如 Black 代表黑色)设置背景。CSS 3 中定义了一些新的属性，设置这些属性还可以控制色调、饱和度、亮度和透明度等信息。

本节介绍 CSS 3 中新增的与颜色有关的属性，包括 HSL、HSLA、RGBA 和 Opacity 四个属性。

10.2.1　HSL属性

HSL 色彩模式是业界的一种颜色标准，它是通过对色调、饱和度、亮度三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色的。HSL 即是代表色调、饱和度和亮度三个通道的颜色，这个标准几乎包括了人类视力所能感知的所有颜色，是目前运用最广的颜色系统之一。

- 色调：即 Hue，0 或者 360 表示红色，120 表示绿色，240 表示蓝色，开发者也可以取其他数值来确定颜色。
- 饱和度：即 Saturation，值在 0%~100%之间。0%意味着灰色，而 100%是全彩。
- 亮度：即 Lightness，它也是一个百分比值，在 0%~100%之间。0%表示黑色，100%表示白色。

【例 10.1】

在本次示例中，用户可以拖动页面中的滑块动态改变<div>标记的背景颜色。实现步骤如下。

步骤 01 向页面中添加 h1 元素，该元素表示文章的标题，接着添加一个 div 元素，该元素中包含多个 p 元素，每一个 p 元素显示一段内容。部分代码如下：

```
<h1>把不喜欢自己的人忘掉</h1>
<div id="MySet">
    <p>    这个世界，总有你不喜欢的人，也总有人不喜欢你。这都很正常。而且，无论你有多好，也无论对方有多好，都苛求彼此不得。因为，好不好是一回事，喜欢不喜欢是另一回事。</p>
    <!-- 省略其他内容 -->
</div>
```

步骤 02 向页面中添加 3 个滑块，这 3 个滑块分别表示 H、S、L 的值。滑块通过 input 元素实现，将 type 属性值设置为 range 即可。另外，还需要为这些 input 元素添加 onChange 事件属性。代码如下：

```
H: <input id="hcolor" type="range" onChange="ChangeColor()" min="0"
  max="360" step="1"/>

S: <input id="scolor" type="range" min="0" max="100" step="1"
  onChange="ChangeColor()"/>

L: <input id="lcolor" type="range" min="0" max="100" step="1" value="40"
  onChange="ChangeColor()" />
```

步骤 03 继续向页面中添加 span 元素，并为该元素指定 id 属性，span 元素用于显示 HSL 的各个取值。代码如下：

```
<span id="showvalue"></span>
```

步骤 04 创建 ChangeColor()函数，在该函数中分别获取 H、S、L 滑块的值，然后再更改 span 元素的文本和 div 元素的背景颜色。函数代码如下：

```
function ChangeColor(){
    var h = document.getElementById("hcolor").value;
    var s = document.getElementById("scolor").value;
    var l = document.getElementById("lcolor").value;

    var pin = h+","+s+"%,"+l+"%";                        //将获取到的值拼接起来
    var value = document.getElementById("showvalue");   //获取 span 对象
    value.innerHTML = "hsl("+pin+")";                    //设置要显示的值
    var set = document.getElementById("MySet");          //获取 div 对象
    set.style.backgroundColor = "hsl("+pin+")";          //指定背景颜色
}
```

步骤 05 页面加载时调用 ChangeColor()函数更改文章内容的背景颜色。代码如下：

```
window.onload = ChangeColor;
```

步骤 06 在浏览器中运行上述代码，查看效果，网页的初始效果如图 10-4 所示。



图 10-4　初始效果

步骤 07 随意拖动图 10-4 中的滑块，改变文章内容的背景颜色，此时效果如图 10-5 所示。



图 10-5　拖动滑块更改背景颜色

10.2.2　HSLA属性

HSLA 是在 HSL 的基础上增加一个透明度的设置，它是 HSL 颜色值的扩展，带有一个 alpha 通道，规定了对象的不透明度。HSLA 的使用语法如下：

```
hsla(hue, saturation, lightness, alpha)
```

在上述语法中，alpha 参数定义不透明度，它的值是在 0~1 之间的浮点数值。其中，0 表示完全透明，1 表示完全不透明。

【例 10.2】

在例 10.1 的基础上进行更改，向用户提供一个动态改变背景色的滑块，拖动时能够动态更改文章内容背景颜色的透明度。实现步骤如下。

步骤 01 向网页中增加一个 range 类型的 input 元素。代码如下：

```
A: <input id="acolor" type="range" min="0" max="1" step="0.1" value="0.5"
  onChange="ChangeColor()" />
```

步骤 02 向 JavaScript 脚本的 ChangeColor()函数中添加代码，获取用户拖动滑块时透明度的值，并且将其添加到指定的变量中。部分代码如下：

```
function ChangeColor(){
    /* 省略其他代码 */
    var a = document.getElementById("acolor").value;
    var pin = h+","+s+"%,"+l+"%,"+a;                  //将获取到的值拼接起来
    var value = document.getElementById("showvalue");  //获取 span 对象
    value.innerHTML = "hsla("+pin+")";                 //设置要显示的值
    var set = document.getElementById("MySet");        //获取 div 对象
    set.style.backgroundColor = "hsla("+pin+")";       //指定背景颜色
}
window.onload = ChangeColor;
```

步骤 03 在浏览器中运行本示例的代码查看效果，透明度为 0.5 时的效果如图 10-6 所示。



图 10-6　透明度为 0.5 时的效果

步骤 04 随意拖动图 10-6 中的滑块更改透明度和色调、饱和度以及亮度等内容，此时效果如图 10-7 所示。



图 10-7　透明度为 1 时的效果

从图 10-7 中可以看出，当前的色调值是 273，饱和度值是 33%，亮度值是 13%，透明度值是 1。

10.2.3 RGBA属性

RGB 色彩模式(也翻译为“红绿蓝”，比较少用)是业界的一种颜色标准，是通过对红(Red)、绿(Green)、蓝(Blue)三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色的，RGB 即是代表红、绿、蓝三个通道的颜色，这个标准几乎包括了人类视力所能感知的所有颜色，是目前运用最广的颜色系统之一。

RGBA 有时候会被描述为一个颜色空间，但是它其实仅仅是 RGB 模型附加了额外的信息。

【例 10.3】

利用前面示例的内容通过 RGBA 更改文章内容的字体颜色。实现步骤如下。

步骤 01 向 HTML 网页中添加 4 个滑块，并且为它们指定 Change 事件。具体代码如下：

```
R: <input id="rcolor" type="range" onChange="ChangeColor()" min="0"
  value="255" max="360" step="1"/>

G: <input id="gcolor" type="range" min="0" max="360" step="1" value="100"
  onChange= "ChangeColor()"/>

B: <input id="bcolor" type="range" min="0" max="360" step="1" value="255"
  onChange="ChangeColor()" />

A: <input id="acolor" type="range" min="0" max="1" step="0.1" value="0.8"
  onChange="ChangeColor()" />
```

步骤 02 添加 ChangeColor()函数，在该函数中首先获取各个滑块的值，然后将这些值拼接起来，再更改 span 元素的值和文章内容的字体颜色。完整代码如下：

```
<script>
function ChangeColor(){
    var r = document.getElementById("rcolor").value;        //获取 Red 对象
    var g = document.getElementById("gcolor").value;        //获取 Green 对象
    var b = document.getElementById("bcolor").value;        //获取 Blue 对象
    var a = document.getElementById("acolor").value;        //获取透明度
    var pin = r+","+g+","+b+","+a;                  //将获取到的值拼接起来
    var value = document.getElementById("showvalue");       //获取 span 对象
    value.innerHTML = "rgba("+pin+")";                      //设置要显示的值
    var set = document.getElementById("MySet");             //获取 div 对象
    set.style.color = "rgba("+pin+")";                      //指定背景颜色
}
window.onload = ChangeColor;
</script>
```

步骤 03 在浏览器中运行上述代码，查看效果，初始效果如图 10-8 所示。

步骤 04 随意拖动滑块更改 rgba()中的值，rgba(72,66,0,0.9)的效果如图 10-9 所示。



图 10-8　初始效果



图 10-9　rgba(72,66,0,0.9)的效果

10.2.4　Opacity属性

在前面两节中，HSLA 和 RGBA 两个属性中都涉及到了透明度的概念，实际上，CSS 3 还为开发者提供了 opacity 属性。opacity 属性声明元素的不透明度，该属性类似于 alpha 透明度，不透明度的值是 0~1 之间的浮点数值。其中，0 表示完全透明，1 表示完全不透明，而 0.5 则表示半透明。

【例 10.4】

重新利用上面的内容演示 opacity 属性的使用。实现步骤如下。

步骤 01　向页面中添加一个改变背景透明度的滑块，默认值是 1，并且为该滑块添加 Change 事件。代码如下：

```
opacity: <input id="opa" type="range" onChange="ChangeOpacity()"
  step="0.1" min="0" value="1" max="1"/>
```

步骤 02　向 ChangeOpacity()函数中添加代码，在这段代码中获取透明度的值，并且更改 ID 属性值是 MySet 的 div 元素的透明度，然后通过 ID 属性值是 showvalue 的 span 元素显示透明度。代码如下：

```
function ChangeOpacity(){
    var opacity = document.getElementById("opa").value;    //获取透明度的值
    var bg = document.getElementById("MySet");     //获取 div 元素
    bg.style.opacity = opacity;                     //设置背景颜色的透明度
    var span = document.getElementById("showvalue");
    span.innerHTML = "当前设置的背景透明度："+opacity;
}
window.onload = ChangeOpacity;
```

步骤 03　在浏览器中运行上述代码查看效果，初始效果如图 10-10 所示。



图 10-10　opacity属性值为 1 时的效果

步骤 04　随意拖动图 10-10 中的滑块查看文章内容透明度的改变情况，如图 10-11 所示显示了透明度为 0.6 时的效果。



图 10-11　opacity属性值为 0.6 时的效果

虽然 HSLA、RGBA 和 opacity 都可以设置透明度，但是它们存在着很重要的区别。opacity 设置元素及其所有子元素的不透明值，但是半透明的 RGBA 或者 HSAL 颜色只会影响在当前元素的声明，对其他元素没有影响。

【例 10.5】

演示 HSLA 和 Opacity 属性的使用，通过该例可以看出它们的不同。实现步骤如下。

步骤 01　向页面中添加两个 div 元素，为每一个 div 元素设置 class 属性，并且第一个 div 元素下都包含一个 p 元素。部分代码如下：

```
<div class="halfopaque">
          这个世界，总有你不喜欢的人，也总有人不喜欢你。这都很正常。而且，无论你有多好，也无论对方有多好，都苛求彼此不得。因为，好不好是一回事，喜欢不喜欢是另一回事。
    <p>      刻意去讨人喜欢，折损的，只能是自我的尊严。不要用无数次的折腰，去换得一个漠然的低眉。纡尊降贵换来的，只会是对方愈发地居高临下和颐指气使。没有平视，就永无对等。</p>
</div>
<div class="halfalpha">
    <!-- 省略其他内容 -->
</div>
```

步骤 02 为上个步骤中的 div 元素指定 CSS 样式，样式代码如下：

```
div .halfopaque{
    background-color: hsl(120,40%,40%);
    opacity: 0.5;
    color: #000000;
    font-size: 14px;
}
div .halfalpha{
    background-color: hsla(120,40%,40%,0.5);
    color: #000000;
    font-size: 14px;
}
```

虽然乍看起来，上面的声明块是相同的，实际上它们之间存在着区别。halfopaque div 元素中的任何文本也都将 50%不透明，即呈现半透明状态；halfalpha div 元素中的文本仍为 100%不透明。

步骤 03 运行上述代码，查看透明效果，如图 10-12 所示。



图 10-12　opacity和hsla的使用

10.3　CSS 3 新增的选择器

要使用 CSS 对 HTML 页面中的元素实现一对一、一对多或者多对一的控制，这就需要用到 CSS 选择器。HTML 页面中的元素就是通过 CSS 选择器进行控制的，CSS 3 新增了多种选择器，下面对这些选择器进行简单了解。

10.3.1 属性选择器

在 CSS 2 中已经引入了属性选择器，属性选择器可以根据元素的属性及属性值来选择元素。CSS 3 中增加了更高级的选择器，这些选择器包含了更多的部分值属性选择器，按照规范的说法，应该将它们称为“子串匹配属性选择器”。

CSS 3 中新增加的 3 个属性选择器及其说明如下。

- [attribute^=“val”]：匹配具有 attribute 属性且值以 value 开头的元素。
- [attribute$=“val”]：匹配具有 attribute 属性且值以 value 结尾的元素。
- [attribute*=“val”]：匹配具有 attribute 属性且值中含有 value 的元素。

【例 10.6】

向页面中添加 3 个 div 元素，并且为这些 div 元素指定不同的 class 属性值。实现步骤如下。

步骤 01 向页面中添加 3 个 div 元素，每一个 div 元素的 class 属性不同。部分代码如下：

```
<div class="testOne">
          这个世界，总有你不喜欢的人，也总有人不喜欢你。这都很正常。而且，无论你有多好，也无论对方有多好，都苛求彼此不得。因为，好不好是一回事，喜欢不喜欢是另一回事。</div>
<div class="Twotest"><!—文章内容 --></div>
<div class="mytestthree"><!-- 文章内容 --></div>
```

步骤 02 定义 class 属性值以 test 开头的 div 元素的背景颜色为#FFAAD5，以 test 结尾的 div 元素的字体颜色为蓝色，字体大小为 14 像素，字体为“仿宋”。步骤如下：

```
div[class^="test"]{
    background-color: #FFAAD5;
}
div[class$="test"]{
    color: blue;
    font-size: 14px;
    font-family: "仿宋";
}
```

步骤 03 在浏览器中运行上述代码，观察属性选择器的作用效果。效果如图 10-13 所示。



图 10-13 属性选择器的效果

10.3.2 结构化伪类选择器

结构化伪类选择器用于向某些选择器添加特殊的效果。在介绍新增的结构性伪类选择器之前，了解一下伪类选择器和伪元素选择器。伪类选择器是指已经定义好的选择器，不能随便取名，例如作用在 a 元素中的 a:link、a:visited、a:hover 和 a:active。伪元素选择器并不是真正的元素使用的选择器，而是针对 CSS 中已经定义好的伪元素使用的选择器，例如 first-line、first-letter、before 和 after 是 4 种伪元素选择器。

CSS 3 中新增加了一些结构化伪类选择器，表 10-1 对这些选择器进行了说明。

表 10-1　结构化伪类选择器

选择器名称	说　明
E:root	匹配文档的根元素。在 HTML 中，根元素永远是 html
E:nth-child(n)	匹配父元素中的第 n 个子元素 E
E:nth-last-child(n)	匹配父元素中的倒数第 n 个结构子元素 E
E:nth-of-type(n)	匹配同类型中的第 n 个同级兄弟元素 E
E:nth-last-of-type(n)	匹配同类型中的倒数第 n 个同级兄弟元素 E
E:last-child	匹配父元素中最后一个 E 元素
E:first-of-type	匹配同级兄弟元素中的第一个 E 元素
E:only-child	匹配属于父元素中唯一子元素的 E
E:only-of-type	匹配属于同类型中唯一兄弟元素的 E
E:empty	匹配没有任何子元素(包括 text 节点)的元素 E

【例 10.7】

更改例 10.6 中的样式代码，指定 class 属性中包含 test 属性值的 div 元素的同级奇数个(例如第 1 个、第 3 个)元素的背景颜色。样式代码如下：

```
div[class*="test"]:nth-of-type(odd){
    background-color: #FFAAD5;
}
```

在浏览器中运行上述代码，查看效果，如图 10-14 所示。



图 10-14　例 10.7 的效果

10.3.3 目标伪类选择器

CSS 3 新增了一种目标伪类选择器，用于匹配相关 URL 指向的 E 元素。

基本语法如下：

```
E:target{
    /*样式代码*/
}
```

如下代码演示了 E:target 选择器的简单使用：

```
p:target{
    border: 2px solid #D4D4D4;
    background-color: #e5eecc;
}
```

10.3.4 UI元素状态伪类选择器

CSS 3 中新增了 4 种 UI 元素状态伪类选择器，其说明如表 10-2 所示。

表 10-2 UI 元素状态伪类选择器

选择器名称	说　明
E:enabled	匹配所有用户界面(form 表单)中处于可用状态的 E 元素
E:disabled	匹配所有用户界面(form 表单)中处于不可用状态的 E 元素
E:checked	匹配所有用户界面(form 表单)中处于选中状态的元素 E
E::selection	匹配 E 元素中被用户选中或处于高亮状态的部分

【例 10.8】

例如，首先向 HTML 页面的表单元素中添加两个单选按钮。代码如下：

```
<form method="post" action="">
    <input type="radio" name="css3" checked="checked" />男
    <input type="radio" name="css3" />女
</form>
```

通过 E:checked 选择器指定选中单选按钮时 input 元素的样式。代码如下：

```
input:checked{
    outline: dashed 5px;
}
```

在浏览器中运行上述代码查看效果。

10.3.5 否定伪类

否定伪类用于匹配所有不匹配简单选择符 s 的元素。基本语法如下：

```
E:not(s):{attribute}
```

例如，下面的样式表示设置非 p 元素的所有元素的背景色：

```
E:not(p){
    background-color: #ff0000;
}
```

10.3.6 通用兄弟选择器

CSS 3 中的通用兄弟选择器是指 E~F，用于匹配 E 元素之后的 F 元素。
基本语法如下：

```
E ~ F : {attribute}
```

例如，下面的样式表示匹配 div 元素之后的 p 元素背景颜色：

```
div ~ p {
    background-color: #00FF00;
}
```

10.4 CSS 3 的新增属性

除了新增颜色、新增选择器外，CSS 3 中还增加了许多属性，使用这些属性可以更加快速地设计网页效果。

10.4.1 边框属性

在 CSS 2 中使用 border 属性只能设置纯色或者简单的线条，CSS 3 中添加了新的边框样式，可以使用图片设置边框样式和颜色，还可以添加阴影框，甚至可以实现创建圆角边框的功能。例如，表 10-3 列出了 CSS 3 中新增的边框属性，并对这些属性进行了说明。

表 10-3 新增加的边框属性

属性名称	说明
border-image	设置边框背景图像
border-image-outset	指定边框图像区域超出边框的量
border-image-repeat	图像边框平铺方式，其值包括 repeated(平铺)、rounded(铺满)和 stretched(拉伸)
border-image-slice	指定图像边框的向内偏移
border-image-source	指定用作边框的图片
border-image-width	指定图片边框的宽度
border-shadow	设置边框的阴影效果
border-radius	设置边框的圆角效果

10.4.2 背景属性

CSS 3 中新增加的背景属性有 3 个，它们分别是 background-clip、background-origin 和 background-size。

- background-clip：指定背景的绘制区域。
- background-origin：指定背景图片的定位区域。
- background-size：指定背景图片的尺寸。

10.4.3 文本属性

CSS 3 中新增加了一系列的文本属性，通常这些属性可以对文本进行操作，表 10-4 列出了这些属性。

表 10-4 新增加的文本属性

属性名称	说 明
hanging-punctuation	指定标点字符是否位于线框之外
punctuation-trim	指定是否对标点字符进行修剪
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色
text-justify	指定当 text-align 设置为 justify 时所使用的对齐方法
text-outline	指定文本的轮廓
text-overflow	指定当文本溢出包含元素时发生的事情
text-shadow	向文本添加阴影
text-wrap	指定文本的换行规则
word-break	指定非中日韩文本的换行规则
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行

10.4.4 盒模型属性

盒模型可以轻松地创建适应浏览器窗口的流动布局或者自适应字体大小的布局。

CSS 3 中增加了一系列的盒模型属性，如表 10-5 所示。

表 10-5 盒模型属性

属性名称	说 明
overflow-x	检索或设置当对象的内容超过其指定宽度时如何管理内容
overflow-y	检索或设置当对象的内容超过其指定高度时如何管理内容
overflow-style	指定溢出元素的首选滚动方法
box-align	定义子元素在盒子内垂直方向上的空间分配方式

续表

属性名称	说　明
box-direction	定义盒子的显示顺序
box-flex	定义子元素在盒子内的自适用尺寸
box-flex-group	定义自适应子元素群组
box-lines	定义子元素分列显示
box-ordinal-group	定义子元素在盒子内的显示位置
box-orient	定义盒子分布的坐标轴
box-pack	定义子元素在盒子内水平方向的空间分配方式

10.4.5 用户界面属性

用户界面属性用来定义与界面有关的内容，表 10-6 对这些属性进行了说明。

表 10-6　用户界面属性

属性名称	说　明
appearance	允许将元素设置为标准用户界面元素的外观
box-sizing	允许以确切的方式定义适应某个区域的具体内容
icon	为创作者提供使用图标化等价物来设置元素样式的能力
nav-down	指定在使用 arrow-down 导航键时向何处导航
nav-index	设置元素的 Tab 键控制次序
nav-left	指定在使用 arrow-left 导航键时向何处导航
nav-right	规定在使用 arrow-right 导航键时向何处导航
nav-up	规定在使用 arrow-up 导航键时向何处导航
outline-offset	对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓
resize	指定是否可由用户对元素的尺寸进行调整
zoom	设置或检索对象的缩放比例

10.4.6 新增的其他属性

CSS 3 中新增加了多种属性，除了前面几节介绍的属性外，还包含多列类布局属性和动画属性等多个属性，表 10-7 对这些属性进行了说明。

表 10-7　CSS 3 中新增的其他属性

属性名称	说　明
columns	可以同时定义多栏的数目和每栏宽度
column-width	可以定义每栏的宽度
column-span	定义元素可以在栏目上定位显示

续表

属性名称	说　明
column-rule	定义每栏之间边框的宽度、样式和颜色
column-rule-color	定义每栏之间边框的颜色
column-rule-width	定义每栏之间边框的宽度
column-rule-style	定义每栏之间边框的样式
column-gap	定义两栏之间的间距距离
column-fill	定义栏目的高度是否统一
column-count	可以定义栏目的数目
column-break-before	定义元素之前是否断行
column-break-after	定义元素之后是否断行
@keyframes	定义动画
animation	所有动画属性的简写属性，除了 animation-play-state 属性
animation-name	定义@keyframes 动画的名称
animation-duration	定义动画完成一个周期所花费的秒或毫秒
animation-timing-function	定义动画的速度曲线
animation-delay	定义动画何时开始
animation-iteration-count	定义动画被播放的次数
animation-direction	定义动画是否在下一周期逆向地播放
animation-play-state	定义动画是否正在运行或暂停
animation-fill-mode	定义对象动画时间之外的状态
transition	简写属性，用于在一个属性中设置 4 个过渡属性
transition-property	规定应用过渡的 CSS 属性的名称
transition-duration	指定过渡效果花费的时间
transition-timing-function	指定过渡效果的时间曲线
transition-delay	指定过渡效果何时开始

提示：CSS 3 的功能非常强大，实际上，它的新增属性远不止本节所介绍的这些，而且本节列出的这些属性并不一定都是常用的，有些属性虽然是 CSS 3 中新增加的，但是并不经常被使用。关于其他的新增属性，感兴趣的读者可以在网络上查找相关的资料。

10.5　实战——以 CSS 3 属性制作漂亮按钮

CSS 3 的功能非常强大，本章简单介绍了 CSS 3 的基础知识和新增属性，本节实战将利用 CSS 3 中的属性制作页面中常见的按钮。图 10-15 显示了第一个页面中的按钮内容。



图 10-15　页面中的漂亮按钮(1)

根据图 10-15 的页面效果设计页面，实现步骤如下。

步骤 01 设计页面标题，标题通过 header 元素实现，向该元素中添加子标题和副标题，并且添加两个链接。页面代码如下：

```
<header>
   <h1>CSS <span>buttons</span></h1>
   <h2>with pseudo-elements</h2>
   <nav class="codrops-demos">
       <a class="current" href="index.html">示例 1</a>
       <a href="index2.html">示例 2</a>
       <a href="index3.html">示例 3</a>
   </nav>
</header>
```

步骤 02 继续在上个页面的基础上添加代码，通过 section 元素控制页面中的按钮。页面代码如下：

```
<section>
   <div id="container buttons">
      <p><a href="#" class="a demo one">Click me! </a></p>
      <p><a href="#" class="a demo one">Come on, don't be afraid</a></p>
      <p><a href="#" class="a demo one">
          You can make this as wide as you want ;) </a></p>
   </div>
</section>
```

步骤 03 为页面中的 div 元素和 a 元素添加 CSS 样式，包括宽度、填充距离、背景、圆角以及阴影效果等内容。代码如下：

```
#container buttons{
    width: 400px;                          //宽度
    padding: 60px 60px 30px;               //填充距离
    border-radius: 20px;                   //圆角样式
    background: rgba(255,255,255,0.3);     //背景
    box-shadow: 3px 3px 3px rgba(0,0,0,0.09) inset;    //阴影效果
    border: 1px solid #CCC;                            //边框样式
}
```

步骤 04 继续为步骤 02 中的 a 元素添加样式，通过 border-radius 属性设置圆角效果，通过 border-shadow 属性设置阴影效果，通过 background-image 属性设置背景图片等。代码如下：

```
.a demo one {
    background-color: #3bb3e0;          //背景颜色
    padding: 10px;                      //填充距离
    position: relative;
    font-family: 'Open Sans', sans-serif;       //字体样式
    font-size: 12px;                    //字体大小
    text-decoration: none;
    color: #fff;                        //字体颜色
    border: solid 1px #186f8f;          //边框
    border-radius: 5px;                 //圆角
    background-image: linear-gradient(bottom, rgb(44,160,202) 0%,
      rgb(62,184,229) 100%);
    box-shadow: inset 0px 1px 0px #7fd2f1, 0px 1px 0px #fff;
    /* 省略浏览器的私有属性设置 */
}
```

步骤 05 设置前两个步骤中其他元素的相关样式代码，具体的代码不再显示。

步骤 06 运行上述代码查看效果，制作的按钮效果如图 10-16 所示。

步骤 07 创建 index2.html 网页，显示一种新类型的按钮，效果如图 10-16 所示。



图 10-16　页面中的漂亮按钮(2)

步骤 08 根据图 10-16 的效果添加页面代码和样式代码，该页面的大多数代码都与上个页面相似，主要不同在于链接按钮的 class 属性。本页面将 class 属性的值指定为 a_demo_two，页面代码不再显示。

步骤 09 为 class 为 a_demo_two 的代码设置样式，部分 CSS 样式如下：

```
.a demo two {
    /* 省略其他内容 */
    color: #fff;
    background-image: linear-gradient(bottom, rgb(44,160,202) 0%,
      rgb(62,184,229) 100%);
    box-shadow: inset 0px 1px 0px #7fd2f1, 0px 6px 0px #156785;
    border-radius: 5px;
}
```

步骤 10　根据需要向页面中添加其他代码。

10.6　本 章 习 题

1. 填空题

(1) __________是代表色调、饱和度和亮度三个通道的颜色。
(2) _________是在 HSL 的基础上增加一个透明度的设置，它是 HSL 颜色值的扩展。
(3) CSS 3 新增加的_________属性用于设置圆角效果。

2. 选择题

(1) opacity 属性的值为__________时表示完全透明。
A. 0　B. 1　C. 0%　D. 100%
(2) CSS 3 中新增加的属性选择器不包括__________。
A. [attribute^="val"]　B. [attribute$="val"]
C. [attribute*="val"]　D. [attribute="val"]
(3) UI 元素状态伪类选择器中，______匹配所有用户界面中处于可用状态的元素。
A. E:enabled　B. E:disabled　C. E:checked　D. E::selection
(4) CSS 3 在实现动画效果时，通过__________定义动画。
A. transform　B. @keyframes　C. animation　D. animation-name

3. 上机练习

细心的读者可能已经注意到，在 10.5 节实战中包含 3 个页面，但是只介绍了前两个页面的功能实现。本次上机练习要求读者利用实战的代码，根据如图 10-17 所示的效果完成页面的最终设计。

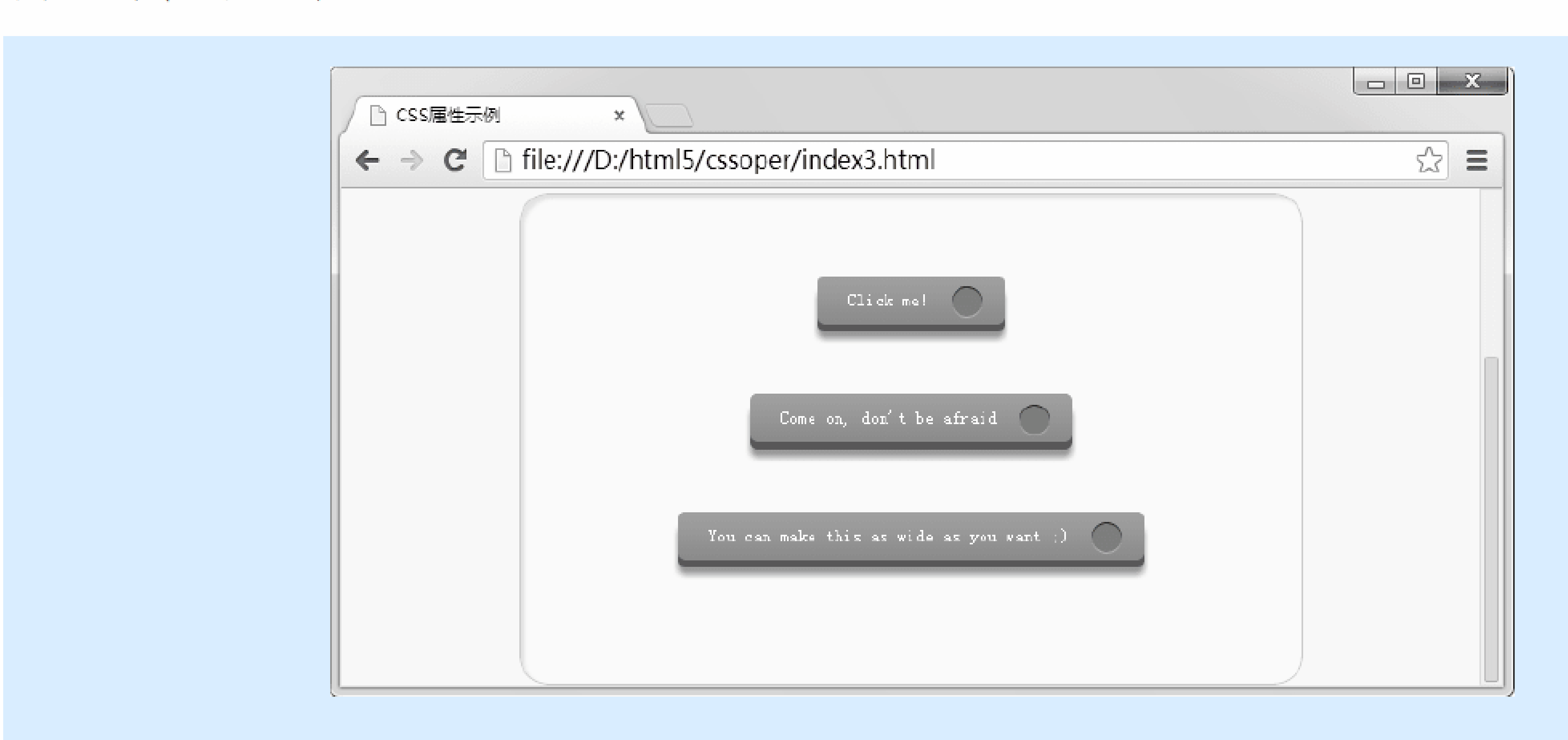


图 10-17　上机练习





CSS 3 新增的选择器

选择器是 W3C 在 CSS 3 工作草案中独立引入的一个新概念，在有些资料中，通常将选择器称为选择符。实际上，CSS 1 和 CSS 2 中已经非系统性地定义了许多常用的选择器(例如元素选择器和样式选择器)，这些选择器基本上能够满足 Web 设计师常规的设计需求。本章向读者介绍 CSS 3 中新增的多种选择器。通过本章的学习，读者可以使用这些选择器设计出漂亮、美观的网页。

本章学习目标：

- 掌握新增的属性选择器
- 掌握 nth 选择器的使用
- 熟悉 E:root 和其他结构化伪类选择器
- 熟悉 E:target 选择器的使用
- 掌握新增的 UI 元素状态伪类选择器
- 熟悉 E:not(s)选择器的使用
- 了解 E~F 选择器的使用

11.1 属性选择器

属性选择器可以根据元素的属性及属性值来选择元素。CSS 3 中新增了 3 种属性选择器，本节简单了解这些选择器。

11.1.1 E[att^=value]选择器

E[att^=value]选择器是指选择匹配 E 的元素，且该元素定义了 att 属性，att 属性值包含前缀为 value 的子字符串。需要注意的是 E 是可以省略的，如果省略时，表示可以匹配任意类型的元素。

【例 11.1】

为了演示本小节以及后面小节介绍的选择器，首先通过普通的 CSS 样式设置显示内容的样式，然后再使用[att^=value]选择器进行替换。实现步骤如下。

步骤 01 向页面中添加 div 元素，并且向该元素中分别添加 h1、div 和 dl 等元素。页面代码如下：

```
<div class="demo clearfix">
    <h1>减肥水果热量排行榜</h1>
    <div>吃水果减肥，是不少女性最喜欢使用的方法之一。那么，生活中常见的一些水果热量到底如何?为了避免盲目选择，不妨让我们给它们排排位:</div>
    <dl>
        <dt><a href="http://www.baidu.com" id="first">1</a>柠檬</dt>
        <dd>100 克/24 卡路里 柠檬中的柠檬酸能促进热量代谢，而且它的维生素 C 含量是水果中的佼佼者，美白效果好，热量又低，爱美想瘦的女性可适量食用，但避免空腹吃。</dd>
        <dt><a href="mailto:abc@163.com" id="second">2</a>菠萝</dt>
        <dd>100 克/32 卡路里 属于酸性水果，可以整肠和助消化，加上菠萝富含的酵素能有益体内毒素分解，促进排水，排脂，所以，那些想要减肥的女性，可以餐后适量食用。</dd>
        <!-- 省略其他内容 -->
    </dl>
</div>
```

步骤 02 为上述内容中的 div 元素和 a 元素添加样式，样式代码如下：

```
.demo {
    width: 95%;
    border: 1px solid #ccc;
    height: 400px;
    padding: 10px;
}
.demo a {
    float: left;
    display: block;
    height: 20px;           /*高度*/
    line-height: 20px;      /*行高*/
    width: 20px;            /*宽度*/
    -moz-border-radius: 10px;          /* Firefox 等浏览器的私有属性 */
    -webkit-border-radius: 10px;       /* Chrome 等浏览器的私有属性 */
    border-radius: 10px;
```

```
    text-align: center;
    background: #f36;          /*背景颜色*/
    color: green;              /*字体颜色*/
    margin-right: 5px;
    text-decoration: none;
}
```

步骤 03 在浏览器中运行上述代码，查看效果，如图 11-1 所示。

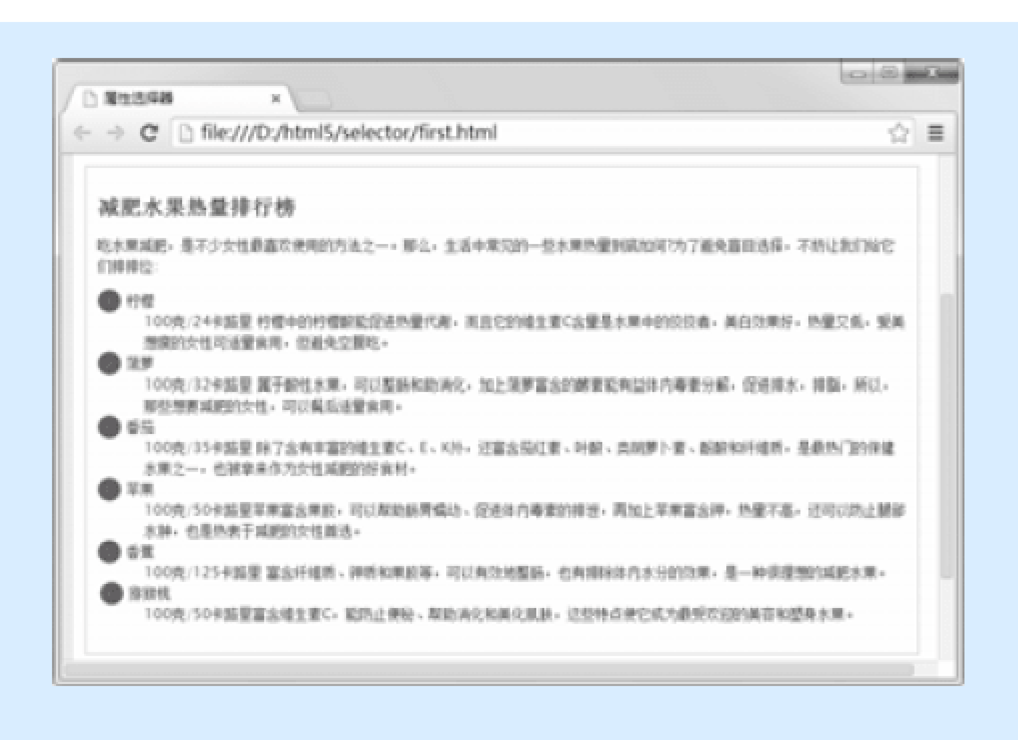


图 11-1 初始运行效果

步骤 04 继续在上面样式的基础上添加新的代码，如下所示：

```
.demo a[href^="http://"]{
    background: orange;
    color: green;
}
.demo a[href^="mailto:"]{
    background: green;
    color: orange;
}
```

上述代码使用到了[att^=value]选择器，该选择器选择了 href 属性，并且分别指定为属性值以“http://”和“mailto:”开头的所有 a 元素。只要 a 元素中的 href 属性值是以“http://”或“mailto:”开头的 a 元素都会选中，并且将以“http://”开头的 a 元素的背景色为 orange，颜色为 green，以“mailto:”开头的 a 元素背景色为 green，颜色为 orange。

步骤 05 刷新或者重新运行上述代码，此时效果如图 11-2 所示。

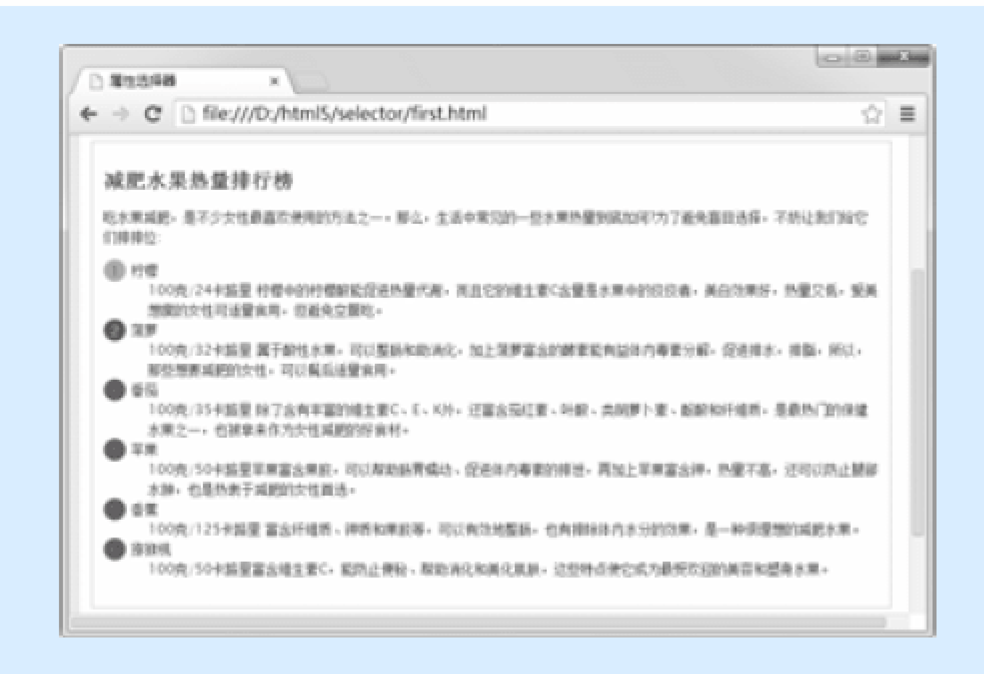


图 11-2 [att^=value]选择器的使用

11.1.2 E[att$=value]选择器

E[att$=value]用于表示选择匹配 E 的元素，且该元素定义了 att 属性，att 属性值包含后缀为 value 的子字符串。E[att$=value]与 E[att^=value]属性器一样，E 元素可以省略，如果省略，则表示可匹配任意类型的元素。

【例 11.2】

继续在前面示例的基础上进行更改，通过 E[att$=value]选择器指定 a 元素的 id 属性，设置 id 属性以 th 结尾的元素样式。相关代码如下：

```
.demo a[id$="th"]{
    background: blue;
    color: white;
}
```

在上述代码中，如果 a 元素的 id 属性以 th 结尾，那么将该元素的背景颜色设置为 blue，字体颜色设置为 white。

在浏览器中运行本例的代码，效果如图 11-3 所示。

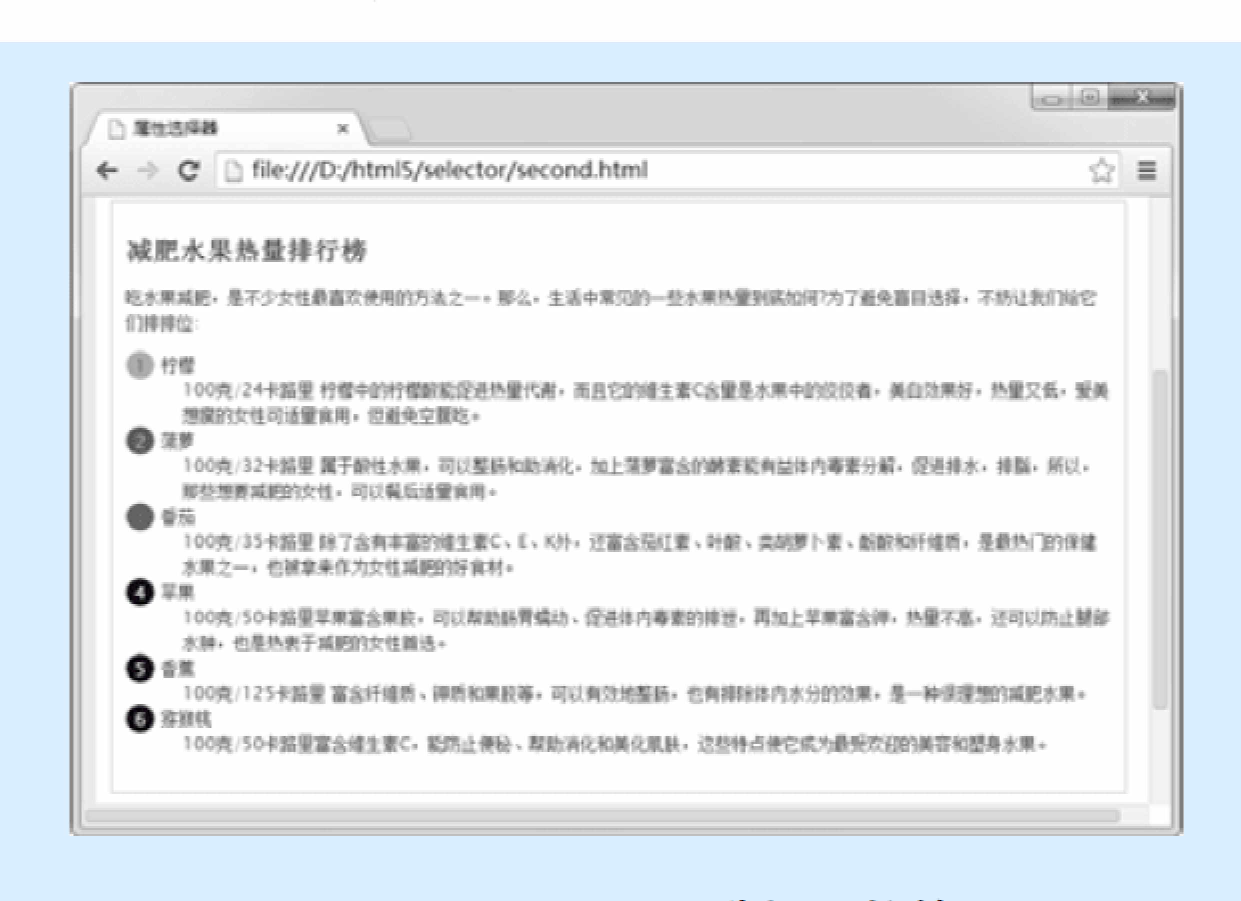


图 11-3　[att$=value]选择器的使用

11.1.3 E[att*=value]选择器

E[att*=value]选择匹配 E 元素，且该元素定义了 att 属性，att 属性值包含 value 的子字符串。该选择器与前两个选择器一样，E 元素也可以省略，如果省略，则表示可以匹配任意类型的元素。

【例 11.3】

在前面示例的基础上进行更改，选择 div.demo 元素下的 a 元素，而元素的 href 属性只要包含#就符合选择条件。代码如下：

```
.demo a[href*="#"]{
    background: black;
    color: white;
}
```

在浏览器中运行本例的代码，查看效果，如图 11-4 所示。



图 11-4　[att*=value]选择器的使用

11.2　结构化伪类选择器

结构化伪类选择器是 CSS 3 选择器的最新部分，它的内容非常重要，有些学者将这节中以 nth 开头的选择器称为 nth 选择器。本节将对 CSS 3 中新增加的结构化伪类选择器进行介绍。

11.2.1　E:root选择器

E:root 选择器很容易理解，它匹配文本的根元素。在一个 HTML 文档中，它的根元素永远是 html，因此 E:root 选择器与 html 元素选择器匹配的内容相同。如下是 E:root 选择器的基本使用：

```
html:root {
    background-color: red;
}
```

11.2.2　E:nth-child(n)选择器

E:nth-child(n)选择器选择所有在父元素中第 n 个位置匹配 E 的子元素。E:nth-child(n)选择器中，参数 n 可以是数字(例如 1、2、3)、关键字(例如 odd 表示奇数、even 表示偶数)、公式(例如 2n、2n+3)，参数的索引起始值为 1，而不是 0。例如 tr:nth-child(3)表示匹配所有表格里排第 3 行的 tr 元素，tr:nth-child(2n)和 tr:nth-child(even)两种形式都表示匹配所有表格的偶数行。

【例 11.4】

在前面示例的基础上更改代码，指定 div.demo 下第 3 个 dt 元素中字体的颜色，并且

指定该元素下 a 元素的圆角边框、背景颜色和字体颜色。样式代码如下：

```
.demo dt:nth-child(3){
    color: #FF8040;
}
.demo dt:nth-child(3) a{
    background: lime;
    color: blue;
    -webkit-border-radius: 0px;                /* Chrome 等浏览器的私有属性 */
    -o-border-radius: 0px;                     /* Opera 等浏览器的私有属性 */
    border-radius: 0px;
}
```

在浏览器中运行本例的代码，效果如图 11-5 所示。



图 11-5　E:nth-child(n)选择器的使用

11.2.3　E:nth-last-child(n)选择器

E:nth-last-child(n)选择器选择所有在其父元素中倒数第 n 个位置的匹配 E 的子元素。该选择器的计算顺序与 E:nth-child(n)相反，但是语法和用法相同。

【例 11.5】

重新更改上个示例中的代码，指定倒数第 2 个 dt 元素的字体颜色和该元素下 a 元素的圆角边框效果。代码如下：

```
.demo dt:nth-last-child(2){
    color: #FF8040;
}
.demo dt:nth-last-child(2) a{
    background: lime;
    color: blue;
    -webkit-border-radius: 0px;                /* Chrome 等浏览器的私有属性 */
    -o-border-radius: 0px;                     /* Opera 等浏览器的私有属性 */
    border-radius: 0px;
}
```

在浏览器中运行上述代码，查看效果，如图 11-6 所示。



图 11-6　E:nth-last-child(n)选择器的使用

11.2.4　E:nth-of-type(n)选择器

细心的用户可以观察例 11.4 和例 11.5，在例 11.4 中指定第 3 个 dt 元素，理想状态下应该更改“番茄”这条记录，可是从图 11-5 中可以看出，更改的是“菠萝”这条记录。在例 11.5 中指定倒数第 3 个 dt 元素，理想状态下应该更改“番茄”这条记录的样式，可是从图 11-6 中可以看出更改的是“猕猴桃”这条记录的样式。出现这种问题的原因是：E:nth-child(n)和 E:nth-las-child(n)选择器在计算子元素是第奇数个元素还是第偶数个元素时，是连同父元素中的子元素一起计算的。以 div.demo dt:nth-child(3)为例，它并不是只针对 div 元素下的第 3 个 dt 元素来使用的，这还包括 div 元素下的其他元素(例如示例中的 dd 元素)。

如果要避免例 11.4 和例 11.5 两种情况的出现，就需要使用到其他的选择器，即 nth-of-type 和 nth-last-of-type 选择器。E:nth-of-type(n)选择器选择父元素中第 n 个位置，且匹配 E 的子元素。在该选择器中，参数 n 的取值与 E:nth-child(n)中的取值一样，可以是数字、关键字或者公式等。E:nth-of-type(n)选择器类似于 E:nth-child(n)选择器，不同的是它只计算选择器中指定的那个元素。

【例 11.6】

重新更改例 11.4 中的代码，通过 E:nth-of-type(n)选择器实现第 3 个 dt 元素中字体的颜色，以及该元素下 a 元素的圆角边框、背景颜色和字体颜色效果。样式代码如下：

```
.demo dt:nth-of-type(3){
    color: #FF8040;
}
.demo dt:nth-of-type(3) a{
    background: lime;
    color: blue;
    -webkit-border-radius: 0px;
    -o-border-radius: 0px;
    border-radius: 0px;
}
```

在浏览器中运行上述代码，查看效果，如图 11-7 所示。



图 11-7　E:nth-of-type(n)选择器的使用

11.2.5　E:nth-last-of-type(n)选择器

E:nth-last-of-type(n)选择器选择父元素中倒数第 n 个位置，且匹配 E 的子元素。它与 E:nth-of-type(n)选择器的语法和用法相同，但是计算顺序与 E:nth-of-type(n)选择器相反。

【例 11.7】

重新更改例 11.5 中的代码，通过 E:nth-last-of-type(n)选择器实现倒数第 2 个 dt 元素及其该元素下 a 元素的效果。样式如下：

```
.demo dt:nth-last-of-type(2){
    color: #FF8040;
}
.demo dt:nth-last-of-type(2) a{
    background: lime;
    color: blue;
    -webkit-border-radius: 0px;              /* Chrome 等浏览器的私有属性 */
    -o-border-radius: 0px;                   /* Opera 等浏览器的私有属性 */
    border-radius: 0px;
}
```

在浏览器中运行本例的代码，观察效果，如图 11-8 所示。



图 11-8　E:nth-last-of-type(n)选择器的使用

11.2.6 E:last-child选择器

E:last-child 选择器选择位于其父元素中最后一个位置，且匹配 E 的子元素。它要比前面几个选择器简单，直接使用即可，不用再传入参数。

【例 11.8】

利用前面示例的页面为元素设计新的样式，指定 div.demo 中最后一个 dd 元素的字体样式、字体颜色和字体大小。

代码如下：

```
.demo dd:last-child{
    font-family: "仿宋";      /*字体样式为仿宋*/
    color: blue;              /*字体颜色为蓝色*/
    font-size: 14px;          /*字体大小为 14*/
}
```

在浏览器中运行上述代码，查看效果，如图 11-9 所示。



图 11-9 E:last-child选择器的使用

11.2.7 E:first-of-type选择器

E:first-of-type 选择器选择在其父元素中匹配 E 的第一个同类型的子元素，该选择器的功能类似于 E:nth-of-type(1)。

【例 11.9】

通过 E:first-of-type 选择器指定 div.demo 中第一个 dd 元素的字体样式、颜色和大小。代码如下：

```
.demo dd:first-of-type{
    font-family: "仿宋";      /*字体样式为仿宋*/
    color: blue;              /*字体颜色为蓝色*/
    font-size: 14px;          /*字体大小为 14*/
}
```

在浏览器中运行上述代码，查看效果，如图 11-10 所示。



图 11-10　E:first-of-type选择器的使用

11.2.8 其他选择器

除了前面介绍的 7 种选择器外，CSS 3 中新增加的结构化伪类选择器还包括 E:only-child、E:only-of-type 和 E:empty。

- E:only-child：选择其父元素只包含一个子元素，且该子元素匹配 E 的元素。
- E:only-of-type：选择其父元素只包含一个同类型的子元素，且该子元素匹配 E 的元素。
- E:empty：选择匹配 E 的元素，且该元素不包含子节点。这里需要注意的是，文本也属于节点。

提示：E:only-child、E:only-of-type 和 E:empty 选择器的使用效果与其他选择器相似，读者可以分别使用这些选择器设置页面中的元素，这里不再具体介绍它们的使用方法。

11.3 目标伪类选择器

E:target 是 CSS 3 中新增加的目标伪类选择器，它选择匹配 E 的所有元素，且匹配元素被相关 URL 指向。E:target 选择器是动态选择器，只有存在 URL 指向该匹配元素时，样式效果才有效。

【例 11.10】

例如，在前面示例的网页中，div.demo 元素下的 a 元素都有一个 id 属性，其属性值分别为 first、second、third、fourth 等。在样式中添加 a:target 的样式，代码如下：

```
a:target{
    background-color: blue;
    outline: 2px solid orange;
    color: white;
}
```

在浏览器中输入 URL 网址，并附加“#third”，以锚点方式链接到<a id=“third”>，则该元素理解显示为蓝色背景、白色字体，如图 11-11 所示。



图 11-11　E:target选择器的使用

11.4　UI 元素状态伪类选择器

UI 元素状态伪类是也是 CSS 3 中新增的全新类型选择器，其中 UI 是 User Interface(用户界面)的简写，UI 设计是指网页的人机交互、操作逻辑、界面美观的整体设计。优秀的 UI 设计不仅是让网页更具个性和品位，还要让网页操作变得便利和简单，充分体现网站的定位和特点。

本节分两小节介绍 CSS 3 中新增的 UI 元素状态伪类选择器，它们分别是 E:enabled、E:disabled、E:checked 以及 E::section 四种。

11.4.1　常用的选择器

E:enabled、E:disabled 和 E:checked 是新增的 3 种 UI 元素伪类选择器。其中，E:enabled 用于选择匹配 E 的所有可用的 UI 元素；E:disabled 用于选择匹配 E 的所有不可用 UI 元素；E:checked 用于选择匹配 E 的所有可用 UI 元素。

在网页中，UI 元素一般是指包含在 form 元素内的表单元素，例如 textarea 元素和 input 元素。

【例 11.11】

在本示例中通过 E:enabled、E:disabled 和 E:checked 元素设置 input 元素在不同状态下的效果。实现步骤如下。

步骤 01　向 HTML 网页的 form 元素中添加一个 4 行两列的表格。内容如下：

```
<form id="form1" action="#" method="get">
    <table width="90%" height="150px" align="center" border="1"
      bordercolor="#CCCCCC" cellpadding="0" cellspacing="0">
        <tr>
            <td align="right" width="20%">登录名：</td>
            <td>
```

```
            <input name="loginname" type="text" value="Hello" disabled />
            </td>
        </tr>
        <tr>
            <td align="right">真实姓名：</td>
            <td><input name="realname" type="text" /></td>
        </tr>
        <tr>
            <td align="right">爱好：</td>
            <td>
                <input name="love1" value="sing" type="checkbox" />唱歌
                <input name="love2" value="dang" type="checkbox" />跳舞
                <input name="love3" value="hua" type="checkbox" />画画
            </td>
        </tr>
        <tr><td></td><td><input type="button" value="提 交"/></td></tr>
    </table>
</form>
```

步骤 02 指定 text 类型的 input 元素可用时的边框样式，内容如下：

```
input[type="text"]:enabled{
    border: 2px solid blue;
}
```

步骤 03 指定 text 类型的 input 元素不可用时的边框样式，内容如下：

```
input[type="text"]:disabled{
    border: 2px solid red;
}
```

步骤 04 指定 checkbox 类型的 input 元素选中时边框边缘的外围样式，内容如下：

```
input[type="checkbox"]:checked{
    outline: 3px dotted #0080FF;
}
```

步骤 05 在浏览器中运行本示例的页面查看效果，初始效果如图 11-12 所示。

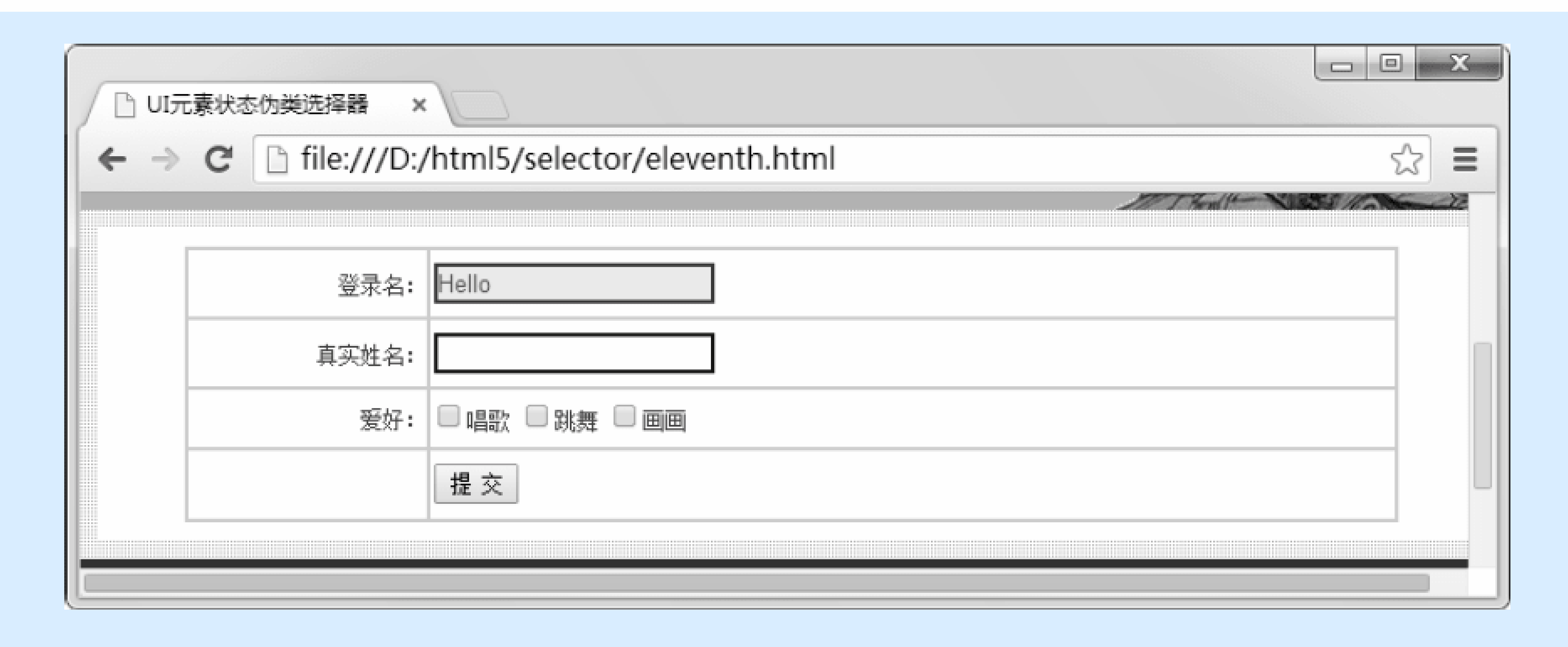


图 11-12　常用的UI元素状态伪类选择器

步骤 06 选中图中与“爱好”有关的复选框，选中时的效果如图 11-13 所示。

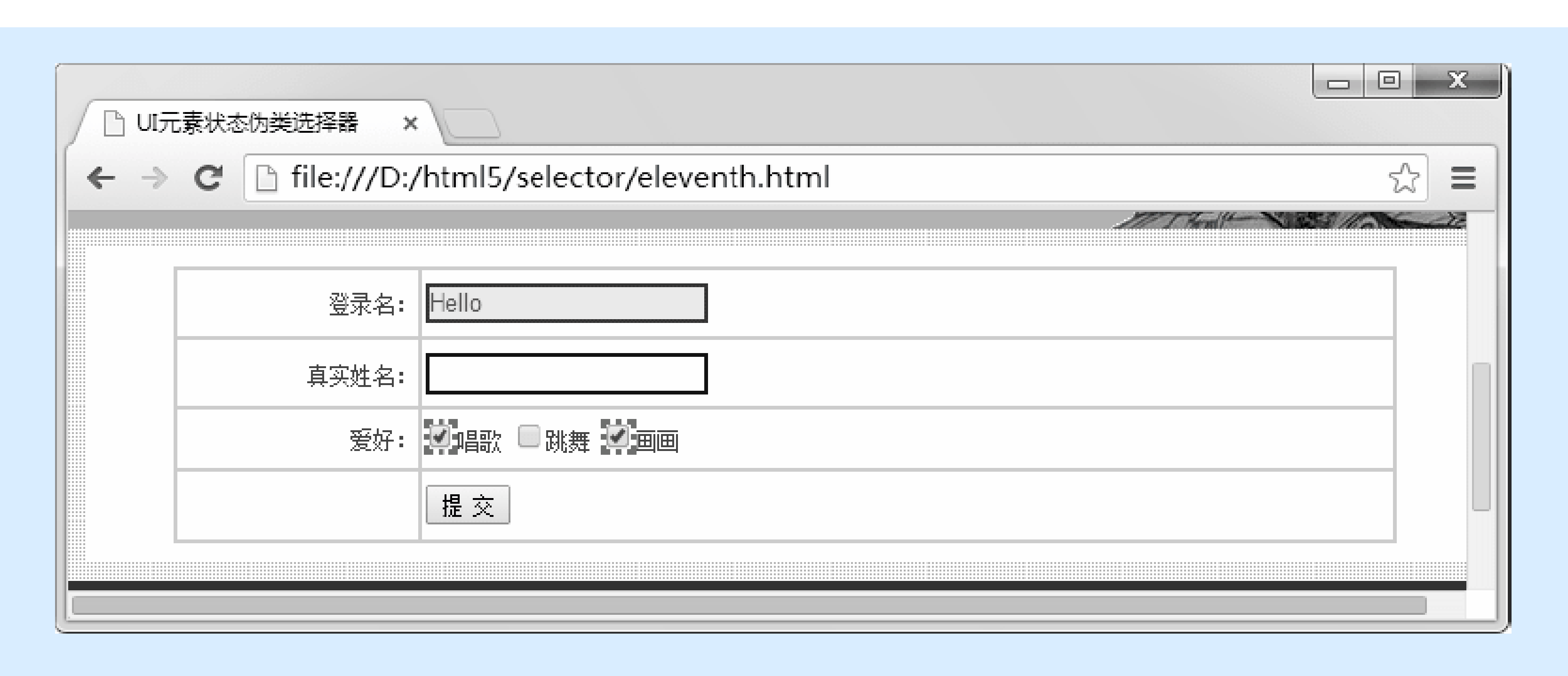


图 11-13　选中复选框时的效果

11.4.2　E::selection选择器

E::selection 选择器用于匹配 E 元素中被用户选中或处于高亮状态的部分。

【例 11.12】

向页面中添加一首歌，包括歌名和歌曲两部分，实现歌词内容选中时的效果。实现步骤如下。

步骤 01　向页面中添加歌曲名称和歌词内容，页面代码如下：

```
<header><h1>特别的爱给特别的你</h1></header>
<p>没有承诺 却被你抓得更紧<br/>没有了你 我的世界雨下个不停<br/>我付出一生的时间
<br/>想要忘记你<br/>但是回忆回忆回忆<br/>从我心里跳出来拥抱你<br/>特别的爱给特别的
你<br/>我的寂寞逃不过你的眼睛<br/>特别的爱给特别的你<br/>你让我越来越不相信自己</p>
```

步骤 02　为 p 元素添加样式代码，用户选中 p 元素中的内容时设置背景颜色为绿色，字体颜色为蓝色。样式如下：

```
p::selection{
    background-color: green;
    color: blue;
}
```

步骤 03　在浏览器中运行本次示例的代码进行查看，选中歌词内容时的效果如图 11-14 所示。

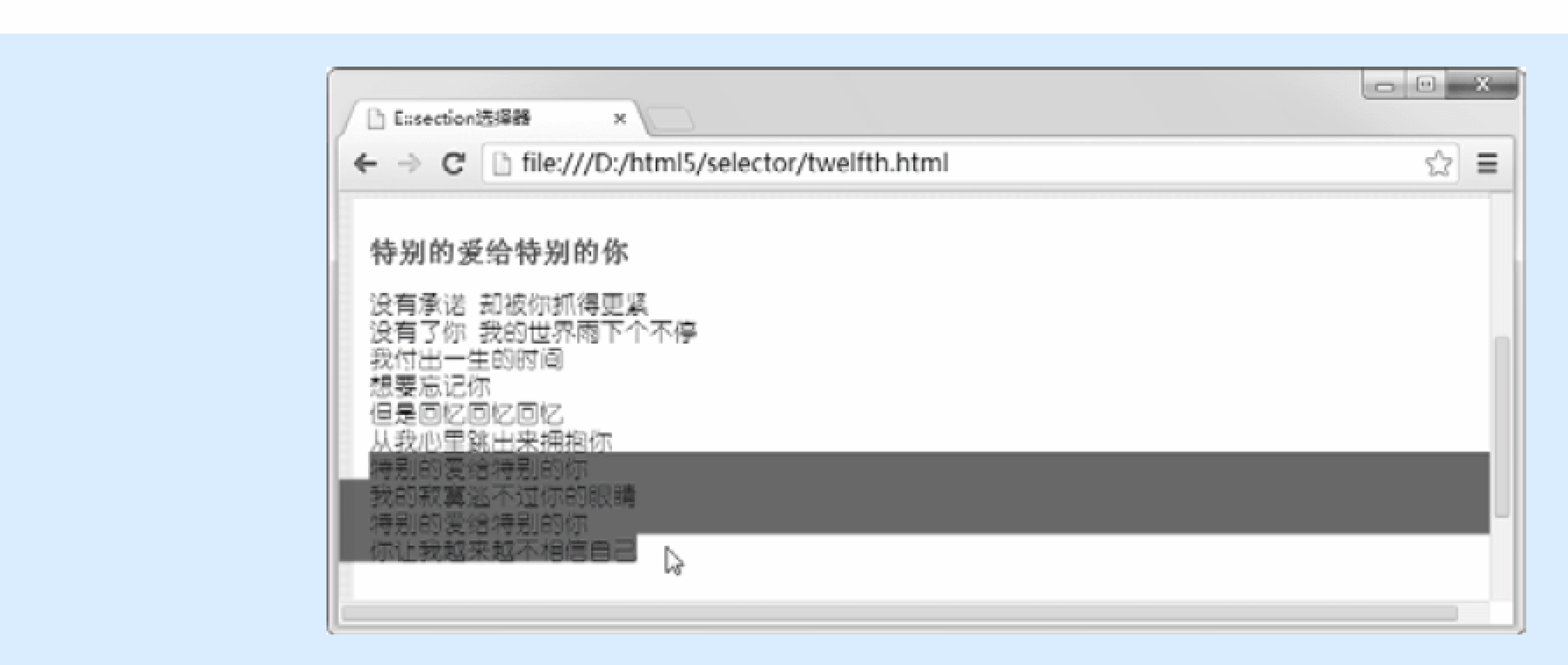


图 11-14　E::selection选择器的使用

11.5　否定伪类选择器

E:not(s)是 CSS 3 中新增的一种否定伪类选择器类型，该选择器选择匹配 E 的所有元素，且过滤掉匹配 s 选择符的任意元素。

在 E:not(s)选择器中，参数 s 只是一个简单结构的选择器，不能使用复合选择器。E:not(s)选择器相当于二次过滤，适合用于精确地选择元素。

【例 11.13】

利用例 11.11 的效果页面练习 E:not(s)选择器的使用，对 form 中所有 input 元素指定边框边缘的外围样式，但是不需要 button 起变化。

代码如下：

```
input:not([type="button"]) {
    outline: 1px solid blue;
}
```

在浏览器中运行上述代码，查看效果，如图 11-15 所示。

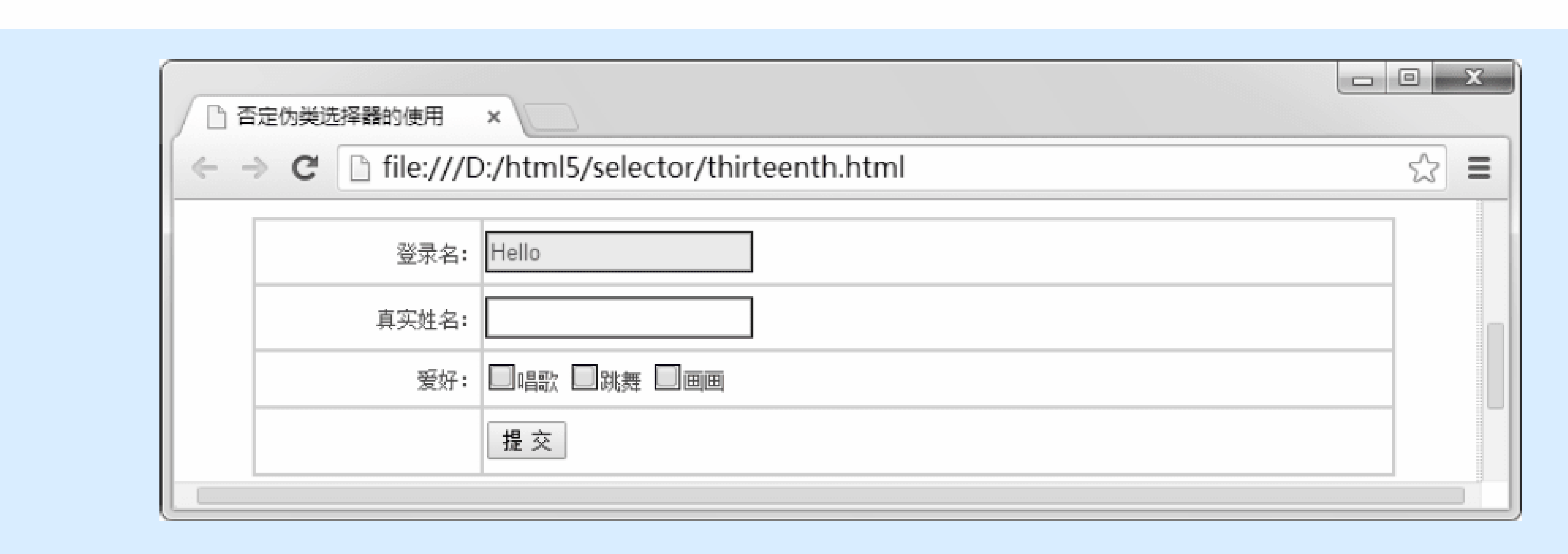


图 11-15　E:not(s)选择器的使用

11.6　通用兄弟选择器

E~F 是 CSS 3 中新增的通用兄弟选择器类型，它用于选择匹配 F 的所有元素，且匹配元素位于匹配 E 的元素后面。需要注意的是，在 DOM 结构树中，E 和 F 所匹配的元素应该位于同一级结构上。

【例 11.14】

在某个网页中包含一个 div 元素，该元素中包含两个 p 元素和一个 header 元素，其中 header 元素位于两个 p 元素之间。

页面代码如下：

```
<div>
    <p>      寒夜的孤独，不是没有人关心没有人
爱，只是我喜欢的那个人她不在。爱你就像呼吸一样理所当然，我可以屏住呼吸，却不能停止爱
你。闭上眼，以为能忘记，但流下的眼泪，却没有骗到自己。一个人一生中究竟要有多少个，才愿
意停留下脚步……
```

```
     </p>
    <header id="myfirst"><h1>北国的思念</h1></header>
    <p>      寒夜思念如殇，在这寂静而又漫长的寒夜
里蔓延开来。寂寞带着文字，直接扎入自己孤独的灵魂。我知道，此生的寂寞，如一缕绵长的丝
纱，在空气里飘荡，蔓延。好想亲近你，感受你温暖的体温，感到你触摸文字的喧哗，我知道，你
喜欢我的文字。犹如这寂寞一样，如痴如醉。可是，想起往日的年华已不在，蔓藤燕莎，大雁飞
去，往事一如飞鸿，让人不敢接近。也许，是习惯了寂寞中独自低吟。如丝般寂寥，感受孤独带来
的斑驳风景，以及倾听寂寞中文字的声音。
    </p>
</div>
```

通过使用 header~p 兄弟选择器为上述代码中的 p 元素指定字体颜色和大小，header~p 表示只为 header 元素之后的 p 元素添加样式。

内容如下：

```
header~p{
    font-size: 14px;
    color: #007DFB;
}
```

在浏览器中运行上述代码，查看效果，如图 11-16 所示。



图 11-16　通用兄弟选择器的使用

11.7　实战——设计直观的表单页面

在本章之前，已经介绍过 CSS 3 中新增的一些选择器，例如属性选择器、UI 元素伪类选择器、目标伪类选择器和否定伪类选择器等。本节实战将前面的内容结合起来，设计出一个直观的表单页面。

设计表单页面时的主要步骤如下。

步骤 01　向页面中添加 form 表单元素，该元素包含一个 7 行两列的表格。页面代码如下：

```
<form id="myform" action="#" method="post">
    <table width="80%" align="center">
        <tr>
            <td>姓名：</td><td><input type="text" required/></td>
        </tr>
```

```
        <tr>
            <td>年龄：</td><td><input type="number" value="18" max="99" />
        </tr>
        <tr>
            <td>户籍所在地：</td>
            <td><input type="text" value="河南 郑州" disabled /></td>
        </tr>
        <tr>
            <td>兴趣：</td>
            <td><input type="checkbox" value="chang" />唱歌
                 <input type="checkbox" value="playbass" />打篮球
                 <input type="checkbox" value="pa" />爬山
            </td>
        </tr>
        <tr>
            <td>性别：</td>
            <td><input type="radio" name="sex" value="1" checked />男
                <input type="radio" name="sex" value="0" />女
            </td>
        </tr>
        <tr>
            <td>备注说明：</td>
            <td><textarea row="20" cols="20"></textarea></td>
        </tr>
        <tr>
            <td> </td><td><input type="button" value="提交" /></td>
        </tr>
    </table>
</form>
```

步骤 02 运行上述页面，查看效果，界面初始效果如图 11-17 所示。

图 11-17　表单初始效果

从图 11-17 中可以看出，本节实战设计的表单页面非常简单，只包含姓名、年龄、户籍所在地、兴趣、性别和备注说明信息。但是，该图中显示的界面不是很美观，例如左侧列的内容与右侧列的内容空白内容过大。

步骤 03 为表单中的元素添加样式，首先为表格的左侧单元格添加样式，指定单元格的背景颜色和字体对齐方式，以及单元格内容选中时的背景色。样式如下：

```
tr>td:first-of-type{
    width: 30%;
    text-align: right;  /* 字体靠右 */
    background-color: #FF7837;  /* 背景颜色 */
}

tr>td:first-of-type::selection{
    background-color: green;
    color: white;
}
```

步骤 04 接着为表格的右侧单元格添加样式，指定单元格的背景颜色。代码如下：

```
tr>td:last-child{
    background-color: #FFAD86;
}
```

步骤 05 为表单中 type 类型是 text 的 input 元素指定圆角边框，并且设置元素不可用时的样式。代码如下：

```
input[type="text"]{
    border-radius: 5px;
}

input[type="text"]:disabled{
    color: red;
    border: 1px solid red;
}
```

步骤 06 为表单中 type 类型是 checkbox 的 input 元素分别指定奇数个和偶数个时边框边缘的外围样式。代码如下：

```
input[type="checkbox"]:nth-of-type(odd){
    outline: 1px dotted blue;
}

input[type="checkbox"]:nth-of-type(even){
    outline: 2px dotted blue;
}
```

步骤 07 为表单中 type 类型是 radio 的 input 元素指定选中时边框边缘的外围样式。代码如下：

```
input[type="radio"]:checked{
    outline: 2px outset red;
}
```

步骤 08 通过:before 选择器在表单之前添加“设计表单页面”文本，并且指定该文本的字体大小和粗细程度。代码如下：

```
table:before{
    content: "设计表单页面";
    font-size: 24px;
    font-weight: bold;
}
```

步骤 09 重新刷新浏览器或者运行页面，效果如图 11-18 所示。



图 11-18 美化表单后的效果

步骤 10 选择表单左侧单元格的内容，观察选中文本的背景颜色，如图 11-19 所示。

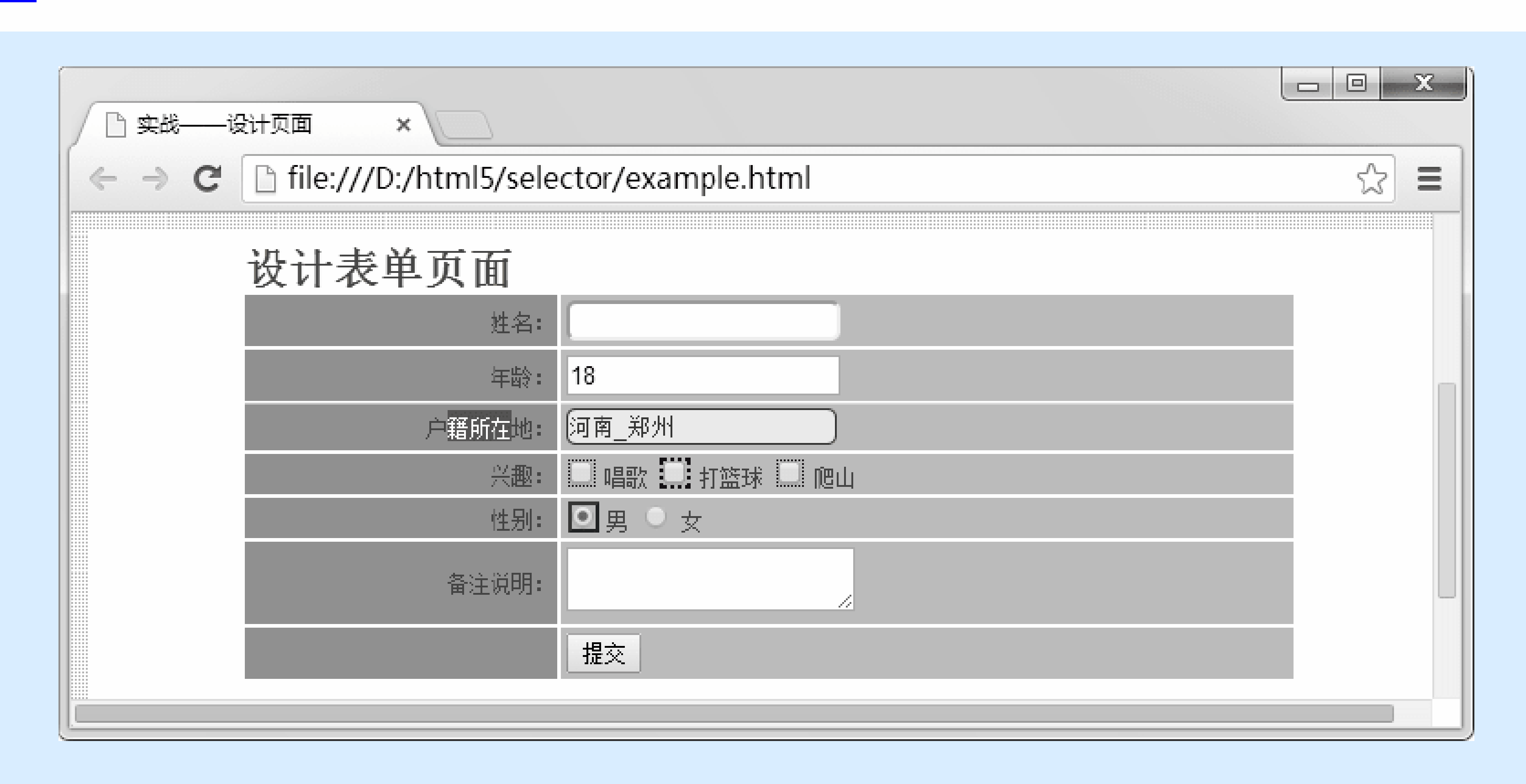


图 11-19 选择左侧单元格的内容

提示： 表单的制作很容易，但是想要设计出友好、易用的表单，就应该考虑多个问题，例如编码格式、样式和结构等。设计表单时应大气、不应捉襟见肘，本节只是为了突出本章重点而设计出一个简单的表单，读者当然可以设计出更漂亮的页面。

11.8 本 章 习 题

1. 填空题

(1) __________选择器是指选择匹配 E 的元素，且该元素定义了 att 属性，att 属性值包含前缀为 value 的子字符串。

(2) __________选择器用于匹配文本的根元素。

(3) 将 E:nth-child(n)选择器的参数 n 设置为__________时，表示选择所有在其父元素

中奇数个位置的匹配 E 的子元素。

(4) __________选择器选择位于其父元素中最后一个位置，且匹配 E 的子元素。

(5) CSS 3 中新增的目标伪类选择器是指__________。

2. 选择题

(1) CSS 3 中新增加的属性选择器不包括__________。

A. E[att=value]　　B. E[att^=value]

C. E[att$=value]　　D. E[att*=value]

(2) __________选择器用于匹配同类型中第 n 个同级兄弟元素。

A. E:nth-child(n)　　B. E:nth-last-child(n)

C. E:nth-of-type(n)　　D. E:nth-last-of-type(n)

(3) CSS 3 新增的__________选择器选择在其父元素中匹配 E 的第一个同类型的子元素，其功能类似于 E:nth-of-type(1)。

A. E:nth-child(1)　　B. E:first-of-type

C. E:only-child　　D. E:only-of-type

(4) 在 CSS 3 新增的 UI 元素状态伪类选择器中，__________用于选择匹配 E 的所有可用的 UI 元素。

A. E:enabled　　B. E:disabled

C. E:checked　　D. E::selection

(5) __________表示否定伪类选择器。

A. E::selection　　B. E~F

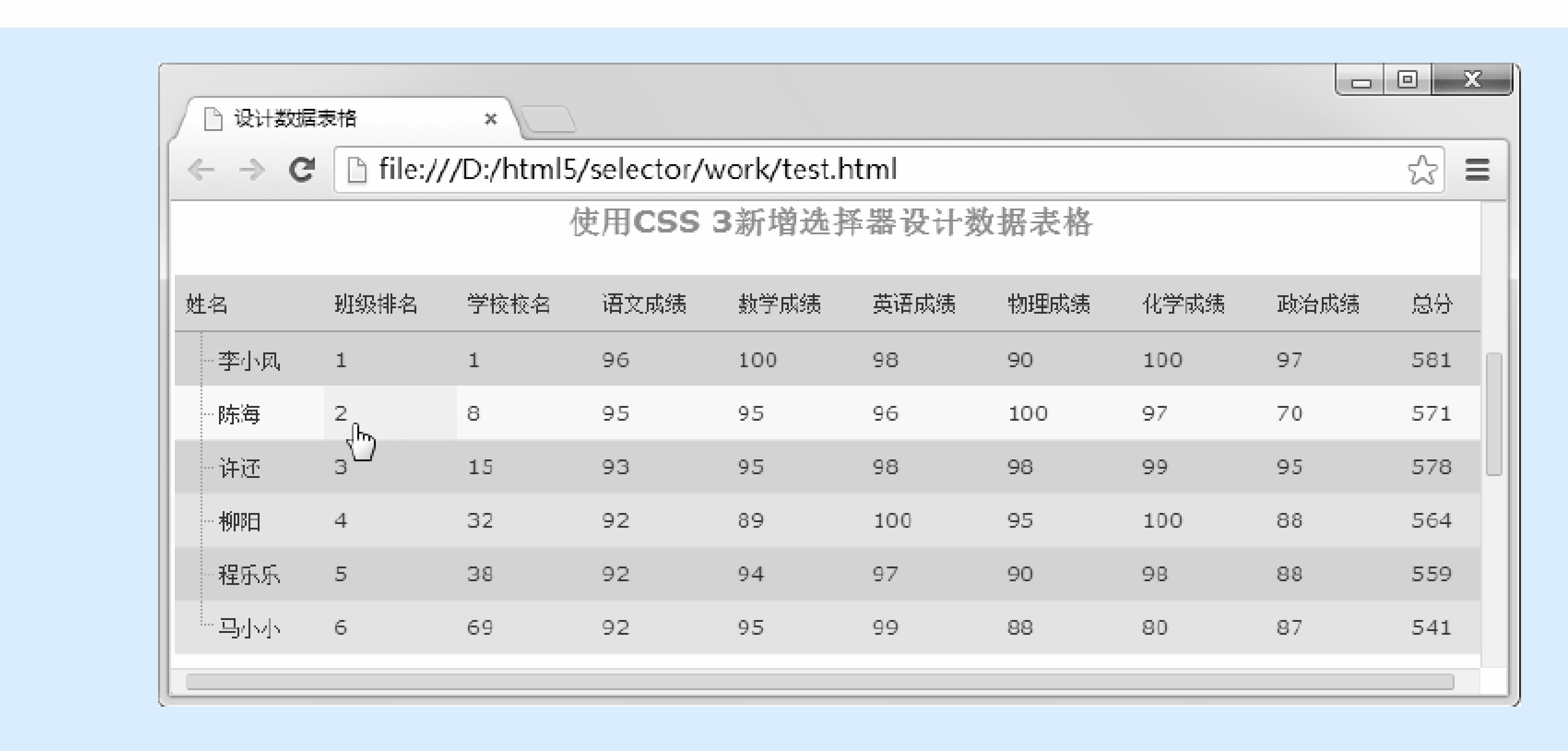
C. E:root　　D. E:not(s)

3. 上机练习

读者需要根据本章介绍的选择器设计层次化的数据表格，表格的初始效果如图 11-20 所示。在图 11-20 中，显示了奇数行(不包括头部标题部分)和偶数行的背景颜色，并且显示了鼠标悬浮至奇数行时的效果。奇数行和偶数行悬浮时的背景颜色不一样，如图 11-21 所示为悬浮到偶数行时的效果。

姓名	班级排名	学校校名	语文成绩	数学成绩	英语成绩	物理成绩	化学成绩	政治成绩	总分
李小风	1	1	96	100	98	90	100	97	581
陈海	2	8	95	95	96	100	97	70	571
许还	3	15	93	95	98	98	99	95	578
柳阳	4	32	92	89	100	95	100	88	564
程乐乐	5	38	92	94	97	90	98	88	559
马小小	6	69	92	95	99	88	80	87	541

图 11-20　表格的初始效果



姓名	班级排名	学校校名	语文成绩	数学成绩	英语成绩	物理成绩	化学成绩	政治成绩	总分
李小风	1	1	96	100	98	90	100	97	581
陈海	2	8	95	95	96	100	97	70	571
许还	3	15	93	95	98	98	99	95	578
柳阳	4	32	92	89	100	95	100	88	564
程乐乐	5	38	92	94	97	90	98	88	559
马小小	6	69	92	95	99	88	80	87	541

图 11-21　悬浮到偶数行时的效果





CSS 3 页面美化样式

CSS 作为样式工具，为页面提供了美化样式工具，包括文本样式、字体样式、边框样式和背景样式等。本章介绍 CSS 新增的页面美化样式。

本章学习目标：

- 了解页面美化的常用样式
- 掌握新增文本样式的用法
- 掌握新增字体样式的用法
- 掌握新增背景样式的用法
- 掌握新增边框样式的用法

12.1　CSS 3 页面美化

页面的美化样式体现在页面的文本样式、字体样式、背景样式和边框样式方面。在 CSS 3 新增的样式中，页面的美化主要体现在如下几点：

- 文本方面新增属性主要表现在标点、换行和对齐的处理；添加了文字的轮廓样式、阴影样式、前景色等样式。
- 字体方面，CSS 3 支持用户使用自定义的字体文件，而并不完全依赖于用户客户端的字体文件。
- 背景方面，CSS 3 新增加了与背景的尺寸、定位和区域相关的属性。
- 边框方面新增了圆角边框、边框背景图片和阴影效果。

上述这些美化效果体现在新增属性方面，在 CSS 3 中通过新增属性来美化页面，使用简单、效果明显。以下分别从文本、字体、背景和边框这几个方面来介绍 CSS 3 中的页面美化。

12.2　文 本 样 式

CSS 3 在不断地更新、强大，而当前浏览器的换代跟不上 CSS 的发展，因此不少 CSS 3 的新增样式无法被支持。本节介绍 CSS 3 的新增文本样式，包括被支持的和不被支持的样式。

12.2.1 新增样式

CSS 3 新增样式都通过属性方式来呈现，对于新增的文本样式，主要体现在如表 12-1 所示的这些属性中。

表 12-1　CSS 3 新增的文本属性

属　性	描　述
hanging-punctuation	规定标点字符是否位于线框之外
punctuation-trim	规定是否对标点字符进行修剪
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色
text-justify	规定当 text-align 设置为 justify 时所使用的对齐方法
text-outline	规定文本的轮廓
text-overflow	规定当文本溢出包含元素时发生的事情
text-shadow	向文本添加阴影
text-wrap	规定文本的换行规则
word-break	规定非中日韩文本的换行规则
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行

上述文本属性并不是所有的浏览器都支持，其中 text-shadow 属性和 word-wrap 属性可支持的浏览器有 Internet Explorer 10、Firefox、Chrome、Safari 以及 Opera。而 Internet Explorer 9 以及更早的版本不支持这些属性。

12.2.2 新增样式的用法

表 12-1 列举了新增的文本属性，本节详细介绍这些属性的用法。由于部分属性并不被当前主流浏览器支持，因此本节重点介绍浏览器支持的属性。

1. text-shadow属性

text-shadow 属性是阴影样式的属性，包含 4 个构成：水平阴影位置、垂直阴影位置、模糊距离和阴影颜色，如表 12-2 所示。

表 12-2 text-shadow 属性构成

属　性	说　明
h-shadow	必需。水平阴影的位置。允许负值
v-shadow	必需。垂直阴影的位置。允许负值
blur	可选。模糊的距离
color	可选。阴影的颜色

定义文本阴影的格式如下：

```
text-shadow: h-shadow v-shadow blur color;
```

【例 12.1】

定义 3 个不同的 div 样式，分别使用不同的阴影参数，查看字体的阴影效果，其样式代码如下所示：

```
<style type="text/css">
    div {
        width: 150px;
        float: left;
        color: #f00;
    }
    .div1 {
        font-size: xx-large;
        text-shadow: 5px 5px 1px #ff00dc;
    }
    .div2 {
        font-size: xx-large;
        text-shadow: 10px 10px 2px #ff00dc;
    }
    .div3 {
        font-size: xx-large;
        text-shadow: 10px 10px 3px #ff00dc;
    }
</style>
```

向页面中添加 3 个 div，分别使用上述 3 种样式，执行效果如图 12-1 所示。

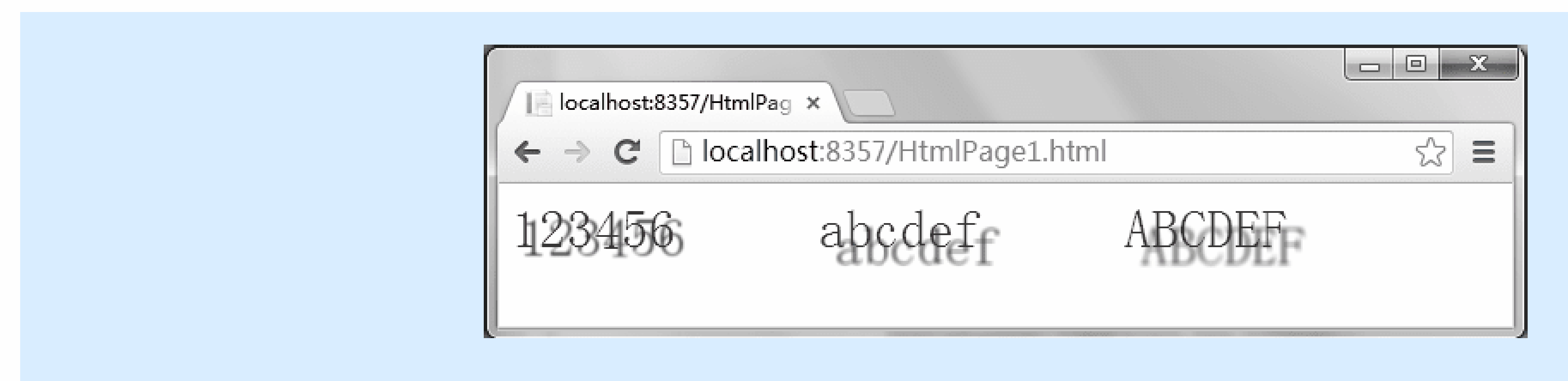


图 12-1　文本阴影

2. word-wrap属性

word-wrap 属性控制文本的换行样式，所有主流浏览器都支持 word-wrap 属性。该属性的语法格式如下所示：

```
word-wrap: normal|break-word;
```

word-wrap 属性有两个参数，对其说明如下所示。

- normal：只在允许的断字点换行(浏览器保持默认处理)。
- break-word：在长单词或 URL 地址内部进行换行。

【例 12.2】

定义两个不同的 div 样式，分别使用相同的宽度和不同的 word-wrap 属性。在页面中添加两个 div，使用这两种样式。两个 div 使用相同长度的文本，分别查看文本中间没有空格和换行的情况、文本中间有空格和换行的情况，操作步骤如下。

步骤 01 首先定义 div 样式，定义 div 共有属性，设置 div 宽度和字体颜色；定义两个 div 样式使用不同的 word-wrap 属性，代码如下：

```
<style type="text/css">
    div {
        width: 150px;
        float: left;
        color: #f00;
    }
    .div1 {
        font-size: xx-large;
        word-wrap: break-word;
    }
    .div2 {
        font-size: xx-large;
        word-wrap: normal;
    }
</style>
```

步骤 02 向页面中添加两个 div，使用相同长度的文本，文本中间没有空格和换行，代码如下：

```
<div class="div1">abcdefghijkl</div>
<div class="div2">ABCDEFGHIJKL</div>
```

由于第一个 div 允许在文本中间换行，因此在长度不够的情况下文本将执行换行，而

不需要考虑文本中间是否有空格或换行符。第二个 div 只允许在空格或换行符处换行，因此即使空间不足，也不能够换行。其执行效果如图 12-2 所示。

步骤 03 修改步骤 02 中的代码，为两个 div 中的文本添加空格，代码如下：

```
<div class="div1">abcdef ghijkl</div>
<div class="div2">ABCDEF GHIJKL</div>
```

对于有了空格的文本，无论是否定义了允许文本中间换行，只要区域内的文本长度大于该区域的宽度，浏览器都将根据空格或换行符进行换行。其执行效果如图 12-3 所示。

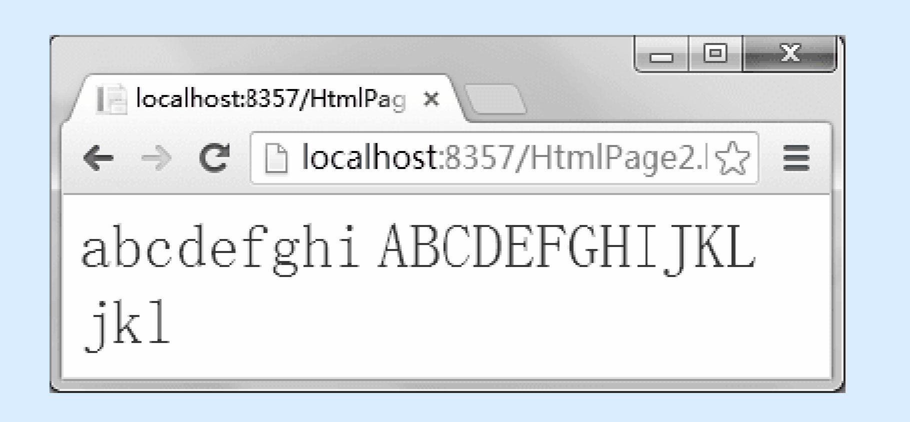


图 12-2　没有断字点的换行

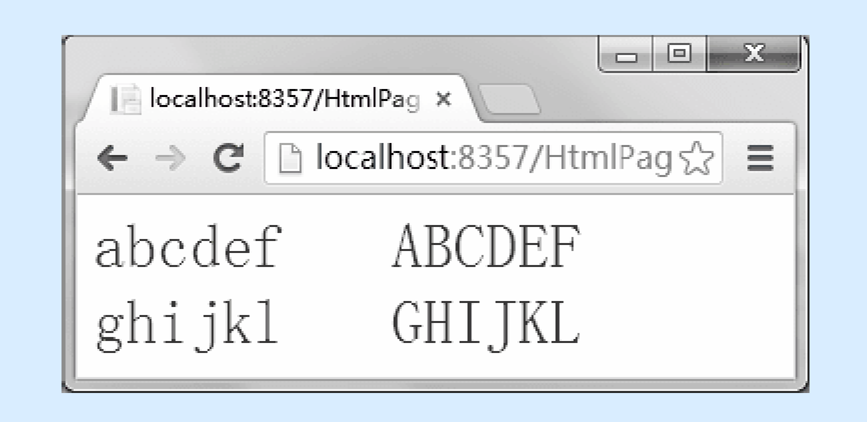


图 12-3　有断字点的换行

3. hanging-punctuation属性

hanging-punctuation 属性规定把标点符号放在文本整行的开头还是结尾的行框之外。目前主流浏览器都不支持 hanging-punctuation 属性，其使用格式如下：

```
hanging-punctuation: none|first|last|allow-end|force-end;
```

上述代码中 none、first、last、allow-end、force-end 为 hanging-punctuation 属性的可取值，其部分属性值的含义如下。

- none：不在文本整行的开头还是结尾的行框之外放置标记符号。
- first：标点附着在首行开始边缘之外。
- last：标点附着在首行结尾边缘之外。

4. punctuation-trim属性

目前主流浏览器都不支持 punctuation-trim 属性。punctuation-trim 属性规定如果标点位于行开头或结尾处，或者临近另一个全角标点符号，是否对标点符号进行修剪。其语法如下所示：

```
punctuation-trim: none|start|end|allow-end|adjacent;
```

上述代码中 none、start、end、allow-end、adjacent 为 punctuation-trim 属性的可取值，其部分属性值的含义如下。

- none：不修剪开启或闭合标点符号。
- start：修剪每行结尾的开启标点符号。
- end：修剪每行结尾的闭合标点符号。

5. text-emphasis属性

text-emphasis 属性是简写属性，用于在一个声明中设置 text-emphasis-style 和 text-

emphasis-color。其语法格式如下：

```
text-emphasis: text-emphasis-style text-emphasis-color;
```

上述代码中，对 text-emphasis-style 属性和 text-emphasis-color 属性的解释如下。

- text-emphasis-style：向元素的文本应用重点标记。
- text-emphasis-color：定义重点标记的前景色。

目前主流浏览器都不支持 text-emphasis 属性。

6. text-justify属性

text-justify 属性改变单词间的间隔，目前只有 Internet Explorer 支持 text-justify 属性。text-justify 属性规定当 text-align 被设置为 text-align 时的齐行方法。该属性规定如何对行文本进行对齐和分隔。其语法如下：

```
text-justify: auto|inter-word|inter-ideograph|inter-cluster
  |distribute|kashida|trim;
```

对上述代码解释如下。

- auto：浏览器决定齐行算法。
- none：禁用齐行。
- inter-word：增加/减少单词间的间隔。
- inter-ideograph：用表意文本来排齐内容。
- inter-cluster：只对不包含内部单词间隔的内容(比如亚洲语系)进行排齐。
- distribute：类似报纸版面，除了在东亚语系中最后一行是不齐行的。
- kashida：通过拉伸字符来排齐内容。

7. text-outline属性

text-outline 属性规定文本轮廓，所有主流浏览器都不支持 text-outline 属性。其语法格式如下所示：

```
text-outline: thickness blur color;
```

对上述代码解释如下。

- thickness：必需。轮廓的粗细。
- blur：可选。轮廓的模糊半径。
- color：必需。轮廓的颜色。

8. text-overflow属性

text-overflow 属性规定当文本溢出包含元素时发生的事情。所有主流浏览器都支持 text-overflow 属性。

其语法格式如下：

```
text-overflow: clip|ellipsis|string;
```

对上述代码解释如下。

- clip：修剪文本。
- ellipsis：显示省略符号来代表被修剪的文本。
- string：使用给定的字符串来代表被修剪的文本。

9. text-wrap属性

text-wrap 属性规定文本的换行(折行)规则。目前主流浏览器都不支持 text-wrap 属性。其语法格式如下所示：

```
text-wrap: normal|none|unrestricted|suppress;
```

对上述代码解释如下。

- normal：只在允许的换行点进行换行。
- none：不换行。元素无法容纳的文本会溢出。
- unrestricted：在任意两个字符间换行。
- suppress：压缩元素中的换行。浏览器只在行中没有其他有效换行点时换行。

10. word-break属性

通过使用 word-break 属性，可以让浏览器实现在任意位置的换行。几乎所有主流浏览器都支持 word-break 属性。

word-break 属性规定自动换行的处理方法，其语法格式如下所示：

```
word-break: normal|break-all|keep-all;
```

对上述代码解释如下。

- normal：使用浏览器默认的换行规则。
- break-all：允许在单词内换行。
- keep-all：只能在半角空格或连字符处换行。

12.3 字 体 样 式

在 CSS 3 之前，Web 设计师必须使用已在用户计算机上安装好的字体。通过 CSS 3，Web 设计师可以使用任意字体。

字体的显示是依赖于用户浏览器的，只有用户浏览器安装过的字体才能够被正常显示。CSS 3 允许将 Web 设计师自定义的(或找到、购买到的)字体存放到 Web 服务器上，它会在需要时被自动下载到用户的计算机上。

自定义的字体是在 CSS 3 的@font-face 选择器中定义的，Firefox、Chrome、Safari 以及 Opera 浏览器支持.ttf(True Type Fonts)和.otf(OpenType Fonts)类型的字体。

Internet Explorer 9 及以上版本支持新的@font-face 规则，但是仅支持.eot 类型的字体(Embedded OpenType)。

在新的@font-face 规则中，必须首先定义字体的名称(例如 myFirstFont)，然后指向该字体文件。

【例 12.3】

如需为 HTML 元素使用字体，通过 font-family 属性来引用字体的名称(myFirstFont)，步骤如下。

步骤 01 首先将该字体在@font-face 选择器中定义，代码如下：

```
@font-face
{
    font-family: myFirstFont;
    src: url('Sansation Light.ttf'),
    url('Sansation Light.eot');
}
```

上述代码中，font-family 属性定义新字体的名称；src 属性定义新字体文件的地址。Windows 目前所支持的字体文件为.ttf、.fon、.ttc 格式。

步骤 02 接着在其他选择器下使用新建字体，如在 div 下使用该字体，代码如下：

```
div
{
    font-family: myFirstFont;
}
```

@font-face 选择器下除了可以定义新建字体的名称和文件地址，还可以定义字体属性，如字体颜色、大小和样式等，如下所示：

```
@font-face
{
    font-family: myFirstFont;
    src: url('Sansation Bold.ttf'),
    url('Sansation Bold.eot');
    font-weight: bold;
}
```

上述代码使用了 myFirstFont 字体，但在定义该字体时使用了加粗属性，该字体在加载时将会显示加粗字体。

由于@font-face 选择器可以定义字体的基础样式(颜色、大小等属性)，因此可将页面中有着相同样式的文本的样式定义为@font-face，并在元素中引用。@font-face 可定义的样式属性如表 12-3 所示。

【例 12.4】

向浏览器中添加行书字体，地址为 Images/xingshu.otf，在 div 中使用该字体显示文本，步骤如下。

步骤 01 首先定义@font-face 选择器添加行书字体，并添加其字体文件；添加 h1 样式，使用行书字体，代码如下：

```
<style type="text/css">
    @font-face {
        font-family: xingshu;
        src: url('Images/xingshu.otf');
    }
    h1 {
        font-family: xingshu;
```

```
    }
</style>
```

表 12-3　@font-face 可定义的样式属性

属性名称	属 性 值	描 述
font-family	name	必需。规定字体的名称
src	URL	必需。定义字体文件的 URL
font-stretch	normal condensed ultra-condensed extra-condensed semi-condensed expanded semi-expanded extra-expanded ultra-expanded	可选。定义如何拉伸字体。默认是 normal
font-style	normal italic oblique	可选。定义字体的样式。默认是 normal
font-size	参照本书第 2 章的表 2-8	可选。设置文本字体大小
font-variant	normal small-caps	可选。设置文本是否大小写
font-weight	normal bold 100~900	可选。定义字体的粗细。默认是 normal
unicode-range	unicode-range	可选。定义字体支持的 Unicode 字符范围。默认是 U+0-10FFFF

步骤 02　向页面中添加 h1，并且添加文字“行书字体”，代码省略。其执行效果如图 12-4 所示。

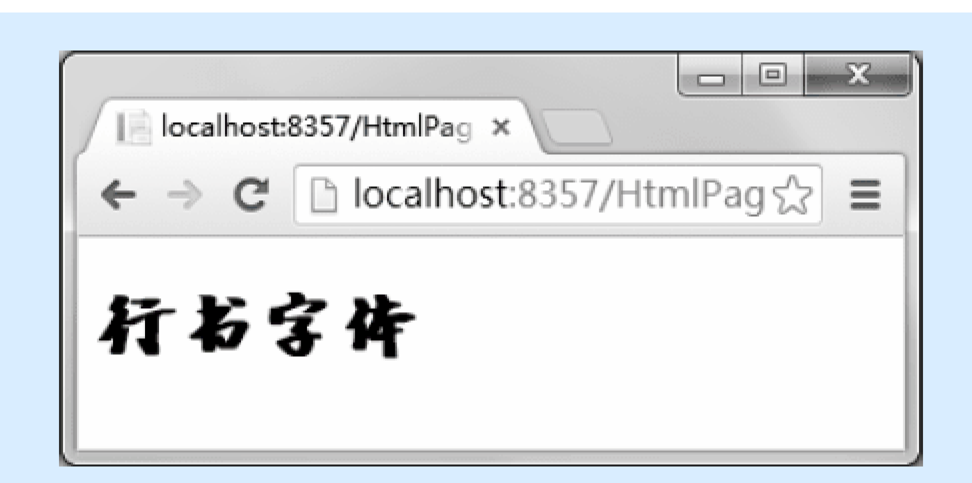


图 12-4　执行效果

12.4 背景样式

CSS 3 包含多个新的背景属性，它们提供了对背景更强大的控制。在 CSS 3 之前，背景图片的尺寸是由图片的实际尺寸决定的。在 CSS 3 中，可以规定背景图片的尺寸，这就允许在不同的环境中重复使用背景图片。CSS 3 允许为元素使用多重背景图像。

开发人员能够以像素或百分比规定尺寸。如果以百分比规定尺寸，那么尺寸相对于父元素的宽度和高度。CSS 3 新增的背景样式主要由以下 3 个属性来控制。

- background-clip：规定背景的绘制区域。
- background-origin：规定背景图片的定位区域。
- background-size：规定背景图片的尺寸。

12.4.1 background-clip属性

CSS 3 的 background-clip 属性设置背景的绘制区域，当前主流浏览器中，IE 9、Firefox、Opera、Chrome 以及 Safari 支持 background-clip 属性。Internet Explorer 8 以及更早的版本不支持 background-clip 属性。其语法格式如下：

```
background-clip: border-box|padding-box|content-box;
```

上述代码中涉及 3 个属性，对其解释如下。

- border-box：背景被裁剪到边框盒。
- padding-box：背景被裁剪到内边距框。
- content-box：背景被裁剪到内容框。

12.4.2 background-origin属性

background-origin 属性规定 background-position 属性相对于什么位置来定位。如果背景图像的 background-attachment 属性为 fixed，则该属性没有效果。

当前主流浏览器中，IE 9、Firefox 4、Opera、Chrome 以及 Safari 5 支持 background-origin 属性。其语法格式如下所示：

```
background-origin: padding-box|border-box|content-box;
```

上述代码中涉及 3 个属性，对其解释如下。

- padding-box：背景图像相对于内边距框来定位。
- border-box：背景图像相对于边框盒来定位。
- content-box：背景图像相对于内容框来定位。

12.4.3 background-size属性

background-size 属性规定背景图像的尺寸。当前主流浏览器中，IE 9、Firefox 4、Opera、Chrome 以及 Safari 5 支持 background-size 属性。其语法格式如下所示：

```
background-size: length|percentage|cover|contain;
```

上述代码中涉及 4 个属性值，对其解释如表 12-4 所示。

表 12-4 background-size 的属性

属 性 值	说 明
length	设置背景图像的高度和宽度。第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为 auto
percentage	以父元素的百分比来设置背景图像的宽度和高度。第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为 auto
cover	把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。背景图像的某些部分也许无法显示在背景定位区域中
contain	把图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域

【例 12.5】

创建页面并添加一个有着两行两列的表格，为表格设置不同的背景样式，查看效果，步骤如下。

步骤 01 首先定义页面表格的共有样式，为了显示 background-clip 属性的效果，为表格定义边框，并统一各个表格的大小，代码如下：

```
td {
   border-style: dashed;
   border-width: thick;
   border-color: #89D4F4;
   width: 300px;
   height: 200px;
}
```

步骤 02 设置背景图片占满表格空间，背景被裁剪到边框盒，使用一个背景图片，代码如下：

```
.td1 {
   background-size: 100% 100%;
   background-image: url('Images/hudie.jpg');
   background-clip: border-box;
}
```

步骤 03 设置背景图片占满表格空间，被裁剪到内边距框，使用一个背景图片，代码如下：

```
.td2 {
   background-size: 100% 100%;
   background-image: url('Images/hudie.jpg');
   background-clip: padding-box;
}
```

步骤 04 设置背景图片占满表格空间，背景被裁剪到内容框，使用一个背景图片，代码如下：

```
.td3 {
    background-size: 100% 100%;
    background-image: url('Images/hudie.jpg');
    background-clip: content-box;
}
```

步骤 05 设置背景图片不重复，使用默认大小、默认布局和默认定位，使用两个重叠的背景图片，第一个是半透明的 PNG 格式图片，第二个是不透明的 JPG 格式图片，代码如下：

```
.td4 {
    background-repeat: no-repeat;
    background-image: url('Images/star.png'),url(Images/hudie.jpg);
}
```

步骤 06 上述 4 个表格使用了相同的背景图片，其运行效果如图 12-5 所示。左上表格图片占据了边框；右上表格图片在表格的边框内部；左下表格图片仅仅占据了表格的内容位置；右下表格图片是正常大小，由于第一个图片是半透明，该表格使用了多重背景图片重叠的样式。

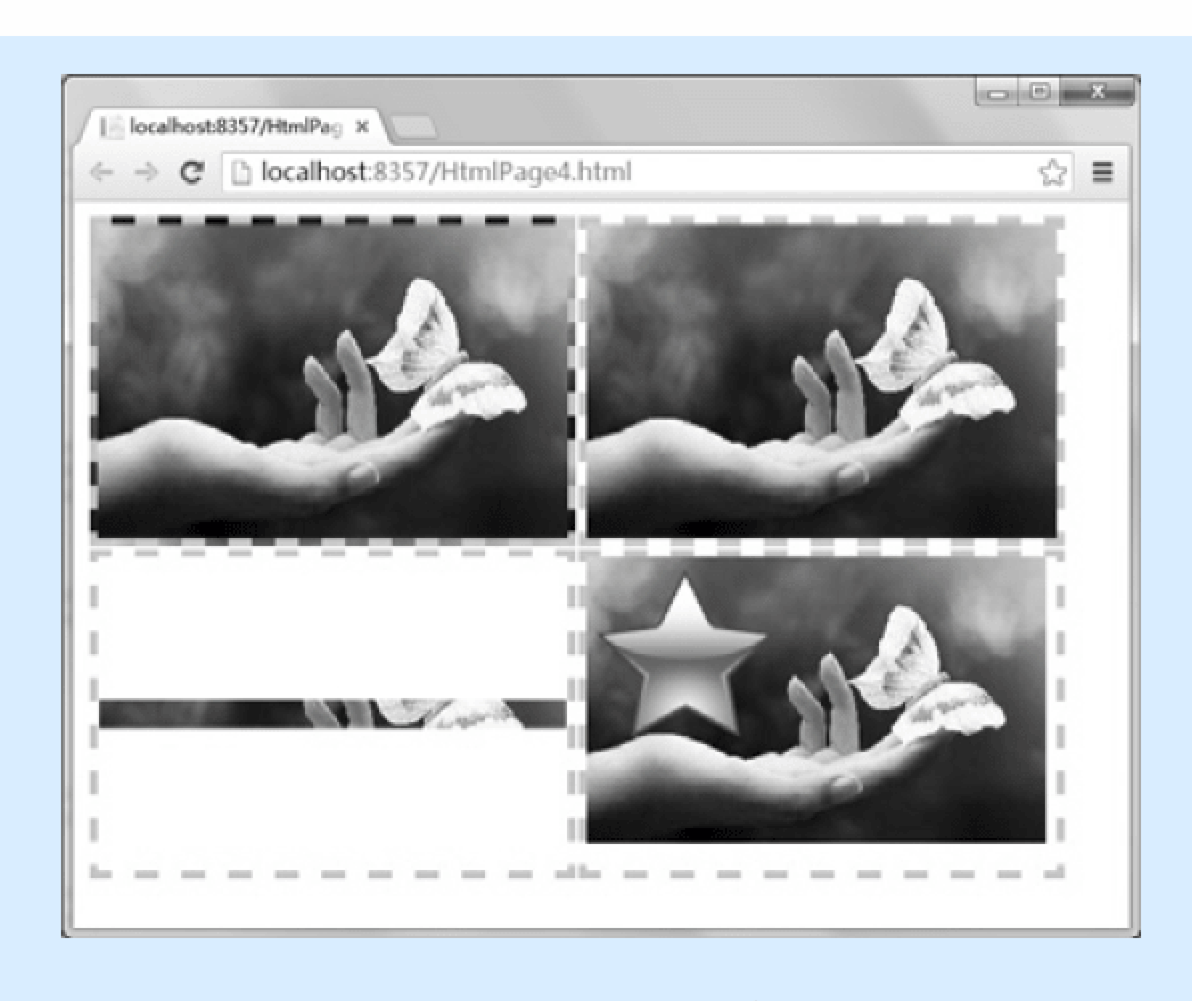


图 12-5　背景样式

12.5　边 框 样 式

CSS 3 中的边框可以使用圆角形状，可以为边框添加阴影，可以使用图片来绘制边框而且不需使用设计软件等。

CSS 3 的边框新增样式主要由 border-radius、box-shadow 和 border-image 属性来控制，其作用如下。

- border-radius：设置边框形状。
- box-shadow：设置边框阴影。
- border-image：设置边框图形。

12.5.1 box-shadow属性

CSS 3 box-shadow 属性向框添加一个或多个阴影。该属性是由逗号分隔的阴影列表，每个阴影由 2~4 个长度值、可选的颜色值以及可选的 inset 关键词来规定。省略长度的值是 0。

当前流行的浏览器中，IE 9、Firefox 4、Chrome、Opera 以及 Safari 5.1.1 支持 box-shadow 属性。其语法格式如下所示：

```
box-shadow: h-shadow v-shadow blur spread color inset;
```

上述 box-shadow 的属性值中，有多个简单属性，这些属性有些是必需的、有些是可选的，如表 12-5 所示。

表 12-5　box-shadow 属性

属　性	说　明
h-shadow	必需。水平阴影的位置。允许负值
v-shadow	必需。垂直阴影的位置。允许负值
blur	可选。模糊距离
spread	可选。阴影的尺寸
color	可选。阴影的颜色
inset	可选。将默认的外部阴影改为内部阴影

【例 12.6】

定义表格的 4 种样式，分别使用不同的 box-shadow 属性，对比 box-shadow 属性的显示效果，步骤如下。

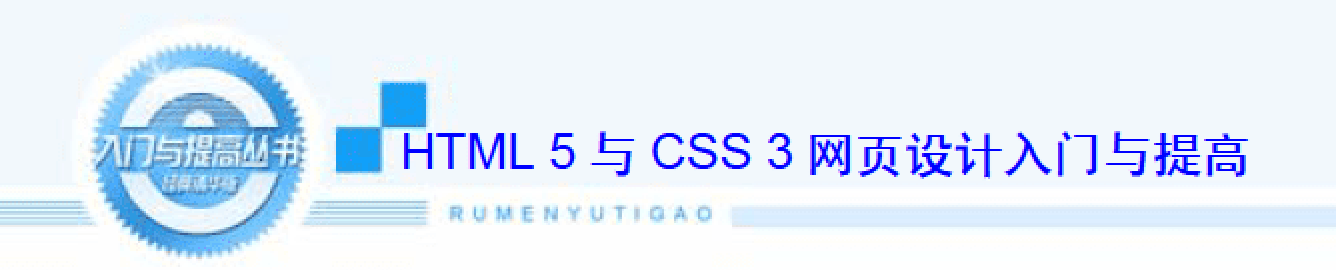
步骤 01　为了对比 box-shadow 属性的显示效果，每一种样式都使用相同的表格样式、粗细和颜色，首先定义 h-shadow、v-shadow、blur、spread 都为 3px 的表格边框，代码如下：

```
.td1 {
    border-style: ridge;
    border-width: thick;
    border-color: #89D4F4;
    box-shadow: 3px 3px 3px 3px red;
}
```

步骤 02　定义 h-shadow、v-shadow、blur、spread 都为 5px 的表格边框，代码如下：

```
.td2 {
    border-style: ridge;
    border-width: thick;
    border-color: #89D4F4;
    box-shadow: 5px 5px 5px 5px red;
}
```

步骤 03　定义 h-shadow、v-shadow、blur、spread 的大小与步骤 01 相同，但显示内



部阴影的边框，代码如下：

```
.td3 {
    border-style: ridge;
    border-width: thick;
    border-color: #89D4F4;
    box-shadow: 3px 3px 3px 3px red inset;
}
```

步骤 04 定义 h-shadow、v-shadow、blur、spread 的大小与步骤 02 相同，但显示内部阴影的边框，代码如下：

```
.td4 {
    border-style: ridge;
    border-width: thick;
    border-color: #89D4F4;
    box-shadow: 5px 5px 5px 5px red inset;
}
```

步骤 05 向页面中添加表格，分别使用上述 4 种样式，这里代码省略。执行效果如图 12-6 所示。



图 12-6　边框阴影

12.5.2　border-image属性

border-image 属性允许设置用于边框的图片。当前浏览器中，Firefox、Chrome 以及 Safari 6 支持 border-image 属性。Opera 支持替代的-o-border-image 属性。Safari 5 支持替代的-webkit-border-image 属性。

border-image 属性是一个简写属性，用于设置多条简单属性，如表 12-6 所示。

表 12-6　border-image 属性

属性名称	说　明
border-image-source	用在边框的图片的路径
border-image-slice	图片边框向内偏移
border-image-width	图片边框的宽度
border-image-outset	边框图像区域超出边框的量
border-image-repeat	图像边框是否应平铺(repeated)、铺满(rounded)或拉伸(stretched)

【例 12.7】

定义 4 种表格样式，分别使用同一个边框背景图片，使用不同的背景显示效果，对比 border-image 属性的显示效果，步骤如下。

步骤 01 首先定义一个只有背景图片，没有背景显示效果的表格样式，以显示该图片的原型，代码如下：

```
.td1 {
   background-image: url('Images/meigui.jpg');
}
```

步骤 02 定义边框样式，使用步骤 01 中的图片设置边框背景，并设置边框的宽度，代码如下：

```
.td2 {
   border-image: url('Images/meigui.jpg');
   border-image-width: 20px;
}
```

步骤 03 在步骤 02 的基础上添加 border-image-slice 属性设置内偏移量，代码如下：

```
.td3 {
   border-image: url('Images/meigui.jpg');
   border-image-width: 20px;
   border-image-slice: 50%;
}
```

步骤 04 在步骤 03 的基础上，修改内偏移量，代码如下：

```
.td4 {
   border-image: url('Images/meigui.jpg');
   border-image-width: 20px;
   border-image-slice: 30%;
}
```

步骤 05 向页面中添加表格，并使用上述 4 种样式，代码省略。执行效果如图 12-7 所示。

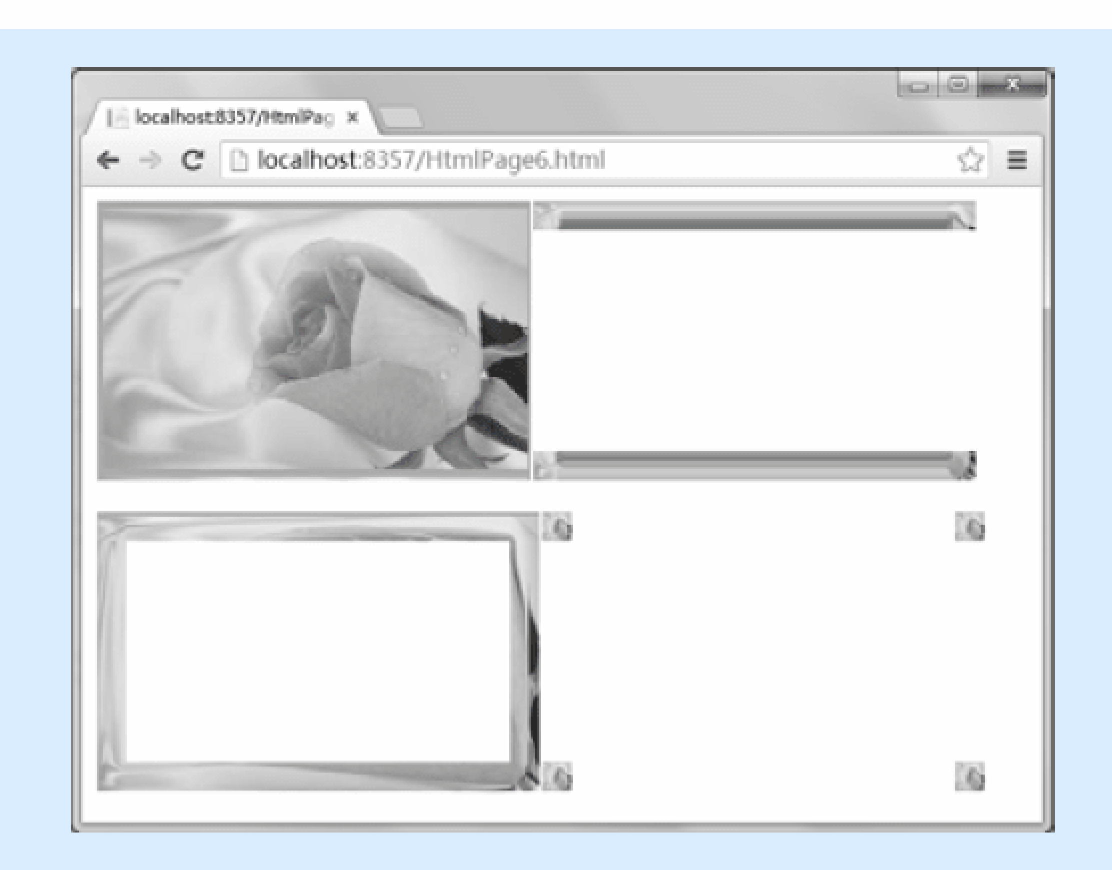


图 12-7　边框背景的使用

12.5.3 border-radius属性

border-radius 属性是一个简单有趣的属性，该属性的取值可以是数值和百分比，通过数值或百分比的大小来改变边框的弧度。

在当前主流浏览器中，IE 9、Firefox 4、Chrome、Safari 5 以及 Opera 支持 border-radius 属性。

同样是通过数值和百分比来定义属性值，border-radius 属性值与其他属性的属性值不同，其百分比和数值分别实现不同的效果：

- 单独使用 4 个角的百分比属性值实现的是不规则边框，其显示效果是左右边框为竖直边框、上下边框为圆弧形边框。
- 单独使用 4 个角的数值属性值实现的是圆角矩形边框，其显示的效果是将边框的 4 个角转换成圆弧的形状，4 个角以外的边框是直线。

上述效果是单独使用 1~4 个数值的显示效果，在 CSS 3 中，可以为边框的每个角设置两个 border-radius 属性值。

无论 border-radius 属性值是百分比还是数值，其所设置的边框形状都与属性值的大小密切相关。

【例 12.8】

分别使用百分比和数值设置两个表的边框，查看其显示效果，表格的样式代码如下：

```
<style type="text/css">
    td {
        border: 2px solid #a1a1a1;
        text-align: center;
        width: 270px;
        height: 80px;
    }
    .td1 {
        border-radius: 30px;
    }
    .td2 {
        border-radius: 50px;
    }
    .td3 {
        border-radius: 30%;
    }
    .td4 {
        border-radius: 50%;
    }
</style>
```

省略表格的添加代码。分别使用上述代码的 4 种边框样式，执行效果如图 12-8 所示。

例 12.8 中为每个角设置了相同的样式，仅仅使用一个属性值来确定表框的每一个角。若为边框设置 4 个值，则表示每一个角的弧度大小。

其实每一个角都有两部分构成，一个是水平弧度；另一个是垂直弧度。以左上角的边框为例，左上角边框相连的有上边框和左边框，那么该角有着与上边框相连的水平弧度和

与左边框相连的垂直弧度，这两种弧度是可以不一样的。

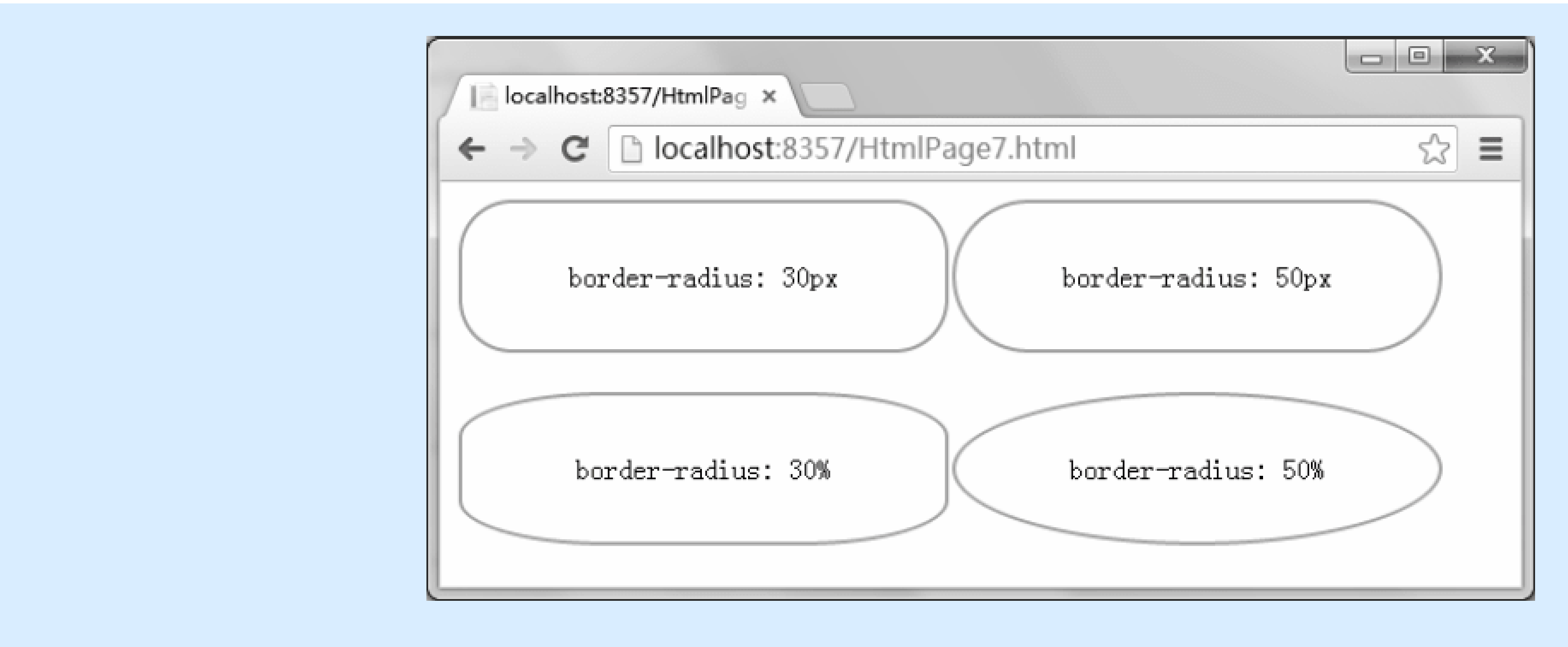


图 12-8　边框角度设置

每一个角的两种弧度之间使用斜线来分隔，如果斜线前后的值都存在，那么斜线前的值设置水平半径，且斜线后的值设置垂直半径。如果没有斜线，则水平半径和垂直半径相等。修改例 12.8，使每个角有着不同的水平弧度和垂直弧度，如例 12.9 所示。

【例 12.9】

在例 12.8 的基础上，为每个角添加不同于水平弧度的垂直弧度，代码如下：

```
.td1 {
    border-radius: 30px/50px;
}
.td2 {
    border-radius: 50px/30px;
}
.td3 {
    border-radius: 30%/50%;
}
.td4 {
    border-radius: 50%/30%;
}
```

这里省略表格的添加代码。分别使用上述代码的 4 种边框样式，其执行效果如图 12-9 所示。

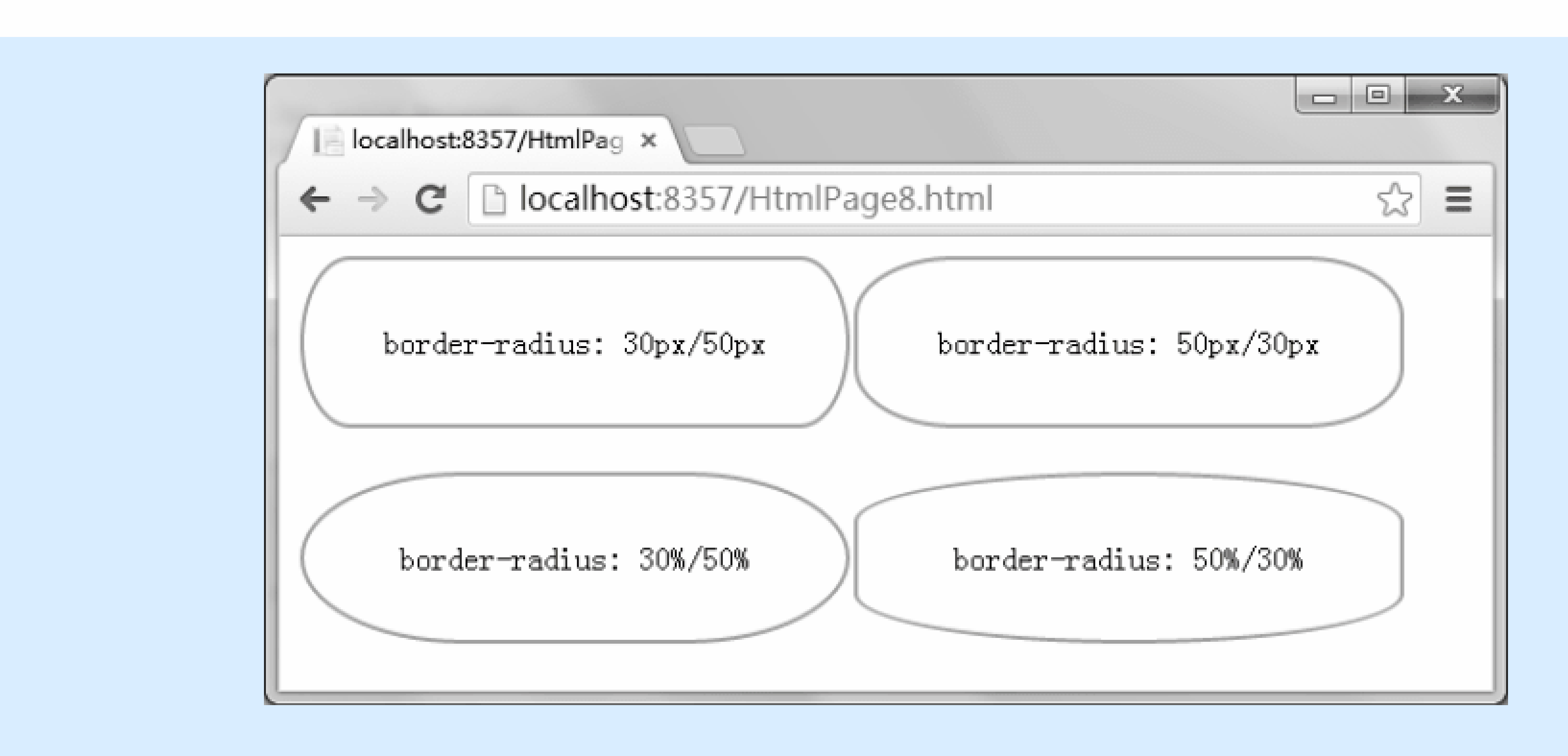


图 12-9　边框形状的水平定义和垂直定义

border-radius 属性是一个简写属性，用于设置边框 4 个角的属性。其定义的顺序依次是左上、右上、左下、右下。

如果省略 bottom-left，则与 top-right 相同。如果省略 bottom-right，则与 top-left 相同。如果省略 top-right，则与 top-left 相同。下列两段代码的效果是一样的。

(1) 简写代码：

```
border-radius: 2em 1em 4em / 0.5em 3em;
```

(2) 完整代码：

```
border-top-left-radius: 2em 0.5em;
border-top-right-radius: 1em 3em;
border-bottom-right-radius: 4em 0.5em;
border-bottom-left-radius: 1em 3em;
```

12.6 实战——表格的艺术

本章详细介绍了 CSS 3 中的新增样式，包括文本样式、字体样式、背景样式和边框样式。根据本章内容，设计不同的表格样式，综合文本样式、字体样式、背景样式和边框样式，要求如下：

- 设计两种表格样式，一种是圆形边框；另一种是图形边框。
- 圆形边框要求表格呈现正规圆形，圆心到边框的距离相等。
- 圆形边框为红色实线，宽度为 40px。
- 圆形边框内部使用有阴影的字体。
- 图形边框使用行书字体。
- 图形边框使用如图 12-10 所示的边框，而内部使用如图 12-11 所示的背景图片。



图 12-10 圆形边框



图 12-11 表格背景

我们来实现上述要求，步骤如下。

步骤 01 首先定义这两个表格的共有属性，两个表格的大小相同，为了能够绘制出正圆形边框，需要使用正方形的表格，即表格的宽和高相等。使用 40px 宽的实线，代码如下：

```
td {
```

```
    border: 40px solid;
    text-align: center;
    width: 160px;
    height: 160px;
}
```

步骤 02 定义圆形表格，呈现正规圆形、圆形边框为红色实线、宽度为 40px、使用有阴影的字体，代码如下：

```
.td1 {
    border-color: #f00;
    background-color: #ffd800;
    border-radius: 50%/50%;
    font-size: xx-large;
    color: red;
    text-shadow: 20px 20px 3px #ff6a00;
}
```

步骤 03 定义图形表格，由于需要使用浏览器没有的字体，因此需要先定义字体样式，代码如下：

```
@font-face {
    font-family: xingshu;
    src: url('Images/xingshu.otf');
}
```

步骤 04 定义图形表格的样式，该表格使用指定的图片作为边框，同时使用指定的图片作为背景。边框图片和背景图片都是.jpg 格式的，没有透明性，因此不能够通过背景多重性进行设置。其设置内容如下所示：

- 分别设置边框背景图片和表格背景图片。
- 设置表格字体为行书字体。
- 设置字体颜色为红色。
- 设置字体大小为 xx-large。
- 设置背景图片的大小为 100%(占满表格)。
- 设置边框宽度能够显示边框图片的边缘。
- 设置边框图片的内偏移能够显示边框图片的边缘。

代码如下：

```
.td2 {
    border-image: url('Images/stars.jpg');
    background-image: url('Images/red.jpg');
    font-family: xingshu;
    color: red;
    font-size: xx-large;
    background-size: 100% 100%;
    border-image-width: 55px;
    border-image-slice: 27%;
}
```

步骤 05 向页面中添加两个表格，使用上述两种样式，显示效果如图 12-12 所示。



图 12-12　表格的艺术

12.7　本 章 习 题

1. 填空题

(1) word-wrap 属性的可取值有 normal 和__________。

(2) 规定背景图片的定位区域的属性是__________。

(3) 边框样式的新增样式主要有 border-radius、__________和 border-image。

(4) 设置边框阴影的属性是__________。

2. 选择题

(1) text-shadow 属性不能够被__________浏览器所支持。

A. IE 9　　B. Chrome　　C. Safari　　D. Opera

(2) 下列不属于新增背景属性的是__________。

A. background-clip　　B. background-origin

C. background-size　　D. background-position

(3) box-shadow 属性值中，不可以省略的属性是__________。

A. blur　　B. v-shadow　　C. spread　　D. color

(4) 下列属性值中，能够被 Opera 支持的是__________。

A. border-image　　B. word-break　　C. box-shadow　　D. text-emphasis

3. 上机练习

网购中，用户根据关键字或商品类型查询商品，系统将给出符合条件的商品信息，布满一个页面。创建商品信息页面，满足如下要求：

- 页面中布满商品信息(图片和说明)。
- 不同品牌的商品使用不同的背景图片边框。
- 特价商品的说明文字使用红色阴影。
- 每一种商品的图片(td 或 div 背景)使用指定的大小。
- 每一种商品的图片(td 或 div 背景)通过相对于内边距框来定位。





第 13 章

CSS 3 页面布局样式

在设计网页时，能否控制好各个模块在页面中的位置是非常关键的。在使用 DIV + CSS 2 进行布局时，提出了盒子模型的概念，它指定了元素在页面上如何显示、显示的位置以及具有的交互功能。这时的盒子主要依赖于 float 属性和 position 属性。

CSS 3 在原来盒子模型的基础上提出了弹性模型，并提供了相关的属性支持。例如，新增的 columns 系列属性可以实现页面的多列布局，box 系列属性可以更加精确地控制子元素的排列位置、对齐方式，以及溢出时的处理方式。除此之外，也新增了一些实用的界面布局属性。

本章将从多列布局、盒模型和界面布局三个方面详细介绍 CSS 3 新增的属性、语法格式及其使用示例。

本章学习目标：

- 熟悉 columns、column-width 和 column-count 属性
- 熟悉 column-gap 属性和 column-rule 属性
- 掌握 column-span 属性和 column-fill 属性
- 熟悉 CSS 3 新增的弹性盒模型的相关属性
- 熟悉 CSS 3 新增的界面布局属性

13.1 新增的多列布局属性

在 CSS 2 以及先前的版本中，依靠浮动布局和定位布局来实现页面的多列布局设计。其中前者比较灵活，但是容易发生布局错乱，从而影响页面的整体效果，而且这种方式需要大量的样式代码，增加了设计人员不必要的工作量；后者可以实现精确定位，但是它无法满足模块的自适用能力，以及模块间的文档流联动需要。

CSS 3 中新增的多列自动布局解决了上面的问题，使用多列布局属性可以自动地将内容按指定的列数排列，非常适合报纸和杂志类网页布局。下面详细介绍多列布局属性的语法及其具体应用。

13.1.1 columns属性

columns 属性用于定义页面上块元素显示的列数，及每列的宽度。该属性的语法格式如下：

```
columns:[column-width] || [column-count]
```

这里的 column-width 和 column-count 又分别是独立的属性。其中，columns-width 用于定义每列的宽度，columns-count 用于定义列数。

【例 13.1】

创建一个示例，使用 columns 属性将博客页面的文章显示分为 3 列。如下所示为文章布局的代码：

```
<div id="middle">
  <h2>最珍贵的东西是免费的</h2>
  <div class="art">
  <br />
  忽然发觉，在这个世界上，最珍贵的东西是免费的。
  <br />
  阳光，是免费的。芸芸众生，没有谁能够离开阳光活下去；然而，从小到大，可曾有谁为自己享受过的阳光支付过一分钱？
  <br />

  <!-- 省略其他内容 -->

  </div>
</div>
```

为上面 class 为 art 的 div 元素添加 CSS 样式。使用 columns 属性定义显示为 3 列，每列宽 150 像素，代码如下：

```
.art {
    columns: 150px 3;
    -moz-columns: 150px 3; /* Firefox */
    -webkit-columns: 150px 3; /* Safari and Chrome */
}
```

在浏览器中运行上述代码，查看多列显示效果，如图 13-1 所示。



图 13-1　多列运行效果

13.1.2　column-width属性

column-width 属性用于设置页面上单列显示的宽度，它适用于除了表格元素之外的非替换块元素、行内块元素和表单格。

column-width 属性的语法如下：

```
column-width: <length> | auto
```

该属性有如下两种取值。

- length：由浮点数字和单位标识符组成的长度值，不可以为负值。
- auto：它根据浏览器计算值自动设置。

提示： column-width 属性可以与其他多列布局属性配合使用，设计指定固定列数、列宽的布局效果；也可以单独使用，限制模块的单列宽度，当超出宽度时，则自动以多列进行显示。

例如，对前面例 13.1 中的样式进行修改，替换成如下使用 column-width 属性的代码：

```
.art {
-moz-column-width: 100px; /* Firefox */
-webkit-column-width: 100px; /* Safari and Chrome */
column-width: 100px;
}
```

再次查看页面，如果网页内容能够在单列内显示，则会以单列显示；如果宽度足够宽，且内容很多，则会在多列中显示，如图 13-2 所示。



图 13-2　columns-width属性的运行效果

13.1.3　column-count属性

column-count 属性用于设置页面上对象显示的列数，它的适用元素与 columns 属性和 column-width 属性一样。语法如下：

```
column-count: <integer> | auto
```

该属性有如下两种取值。

- integer：用来定义栏目的列数，它的取值是一个大于 0 的整数，不允许有负值。如果 column-width 和 column-count 属性没有明确值，则该值为最大列数。
- auto：根据浏览器计算值自动设置。

例如，对例 13.1 中的样式进行修改，替换成如下使用 column-count 属性的代码：

```
.art {
-moz-column-count: 3; /* Firefox */
-webkit-column-count: 3; /* Safari and Chrome */
column-count: 3;
}
```

再次查看页面，无论浏览器窗口怎么调整，页面总是会显示为 3 列，如图 13-3 所示。



图 13-3　column-count属性的运行效果

13.1.4 column-gap属性

column-gap 属性用于设置多列布局时的列间距，语法如下：

```
column-gap: <length> | normal
```

取值有两个：length 表示由浮点数字和单位标识符组成的长度值，不可以为负值；normal 根据浏览器默认设置进行解析，一般为 1em，即 16px。

【例 13.2】

例如，例 13.1 的例子重新设计成显示 3 列，每列之间距离为 30px。样式代码如下：

```
.art {
-moz-column-count: 3;        /* Firefox */
-webkit-column-count: 3;     /* Safari and Chrome */
column-count: 3;
-moz-column-gap: 30px;      /* Firefox */
-webkit-column-gap: 30px;  /* Safari and Chrome */
column-gap: 30px;
}
```

在浏览器中运行页面，本次练习的代码效果如图 13-4 所示。



图 13-4　column-gap属性的运行效果

13.1.5 column-rule属性

column-rule 属性用于设置多列布局时列之间边框的宽度、样式和颜色。语法如下：

```
column-rule: [column-rule-width]|[column-rule-style]|[column-rule-color]
```

各个属性值的含义如下。

- column-rule-width：设置列之间的边框宽度。
- column-rule-style：设置列之间的边框样式。
- column-rule-color：设置列之间的边框颜色。

【例 13.3】

在例 13.2 的基础上进行修改，设置列之间的边框为 2 像素，使用白色虚线显示。修改后的样式代码如下：

```
.art {
    -moz-column-count: 3;  /* Firefox */
    -webkit-column-count: 3; /* Safari and Chrome */
    column-count: 3;
    -moz-column-gap: 30px; /* Firefox */
    -webkit-column-gap: 30px;  /* Safari and Chrome */
    column-gap: 30px;
    -webkit-column-rule: 2px dashed #FFF;   /* 设置列与列之间的边框样式 */
    -moz-column-rule: 2px dashed #FFF;
    column-rule: 2px dashed #FFF;
}
```

在浏览器中运行本次练习的代码，观察效果，如图 13-5 所示。



图 13-5　column-rule属性的运行效果

column-rule 复合属性派生出了三个与列边框相关的属性：column-rule-width 属性、column-rule-style 属性和 column-rule-color 属性。

- column-rule-width：用于设置列与列之间的边框宽度。值是浮点数，但是不能为负值；如果值为 none，则自动忽略该属性。
- column-rule-style：用于设置列与列之间的边框样式，如果 column-rule-width 属性的值设置为 0，则自动忽略该属性。该属性的取值可以是 none、hidden、dotted、dashed、solid、double、groove、ridge、inset 和 outset。
- column-rule-color：用于设置列与列之间边框的颜色，值可以是所有的颜色。如果 column-rule-width 等于 0，或 column-rule-style 设置为 none，本属性将会自动地被忽略。

例如，例 13.3 中的样式代码可以使用以下代码来代替：

```
.art {
    column-rule-width: 2px;            /* 边框宽度 */
    column-rule-style: dotted;         /* 边框样式 */
    column-rule-color: #FFF;           /* 边框颜色 */
    /* 省略兼容浏览器时私有属性的设置和其他样式设置 */
}
```

13.1.6 column-span属性

column-span 属性用于设置对象元素是否横跨所有列，它适用于除浮动和绝对定位之外的块级元素。基本语法如下：

```
column-span: none | all
```

column-span 属性的取值非常简单，none 表示不跨列，只在本栏中显示；all 则表示横跨所有列，并定位在列的 Z 轴之上。

在前面的示例中，文章标题是通过 center 来居中显示的。除了这种方式外，可以使用 solumn-span 属性实现相同的效果。样式代码如下：

```
#center content h2{
    text-align: center;
    -webkit-column-span: all;
    -moz-column-span: all;
    column-span: all;
    /* 省略其他属性的设置 */
}
```

13.1.7 column-fill属性

column-fill 属性用于设置所有列的高度是否统一，适用于多列布局元素。

其语法如下：

```
column-fill: auto | balance
```

取值有两种：auto 表示列高度自适应内容，各列的高度随其内容的变化而自动变化；balance 表示所有列的高度以其中最高的一列进行统一。

如下代码展示了 column-fill 属性的应用：

```
.test{
    width: 600px;
    border: 10px solid #000;                /* 边框样式 */
    column-count: 2;                        /* 显示的列数 */
    column-gap: 20px;                       /* 间隙大小 */
    column-rule: 3px solid #090;            /* 列之间的边框样式 */
    column-fill: balance;
    /* 省略兼容浏览器时私有属性的设置 */
}
```

13.2　新增的盒模型属性

CSS 3 对原来的盒模型进行了调整，提出了弹性盒模型的概念。新的弹性盒模型可以更加灵活地决定元素在盒子中的分布方式以及如何处理盒子的可用空间。

CSS 3 中与弹性盒模型相关的有 8 个属性，下面详细介绍。

13.2.1 box-orient属性

box-orient 属性用于设置弹性盒模型中子元素的排列方式。

box-orient 属性的语法如下：

```
box-orient: horizontal | vertical | inline-axis | block-axis
```

如下对 box-orient 属性的取值进行了说明。

- horizontal：设置弹性盒模型对象中子元素为水平排列，即盒子元素从左到右在一条水平线上显示它的子元素。
- vertical：设置弹性盒模型对象的子元素为纵向排列，即盒子元素从上到下在一条垂直线上显示它的子元素。
- inline-axis：盒子元素沿着内联轴显示它的子元素。
- block-axis：盒子元素沿着块轴显示它的子元素。

注意：在使用弹性盒模型的时候，需要先把父容器的 display 属性设置为 box 或者 inline-box。

【例 13.4】

在 CSS 2 盒模型中，HTML 文档流总是在垂直方向上排列盒子。使用弹性盒模型就可以重新定义文档流的顺序。

假设在一个 HTML 页面中有如下布局代码：

```
<div class="art">
    <p>最珍贵的东西是免费的</p>    <p>烟民的十五个等级</p>
    <p>世界最伟大的管理原则</p>    <p>一个网页设计师的情书</p>  <p>音乐心情</p>
  </div>
```

上述代码中，class 为 art 的 div 是一个父容器，其中包含了 5 个 p 元素。使用 CSS 定义 p 元素的样式，代码如下：

```
.art p{
    margin: 5px;
    padding: 5px;
    line-height: 50px;
    text-align: center;
    border: 1px #000 solid;
    background-color: #F5F5F5;
}
```

在浏览器中查看效果，此时 5 个 p 元素将会垂直排列在 art 容器内，如图 13-6 所示。

下面为 art 容器添加 box-orient 属性，使子元素呈水平方向显示，样式代码如下：

```
.art {
    display: -webkit-box;
    -webkit-box-orient: horizontal;            /* Webkit 引擎 */
    display: -moz-box;
    -moz-box-orient: horizontal;               /* Mozilla Gecko 引擎 */
```

```
    display: box;
    box-orient: horizontal;                    /* 兼容标准 */
}
```

再次在浏览器中查看效果，会发现 5 个 p 元素水平显示在 art 容器内，效果如图 13-7 所示。

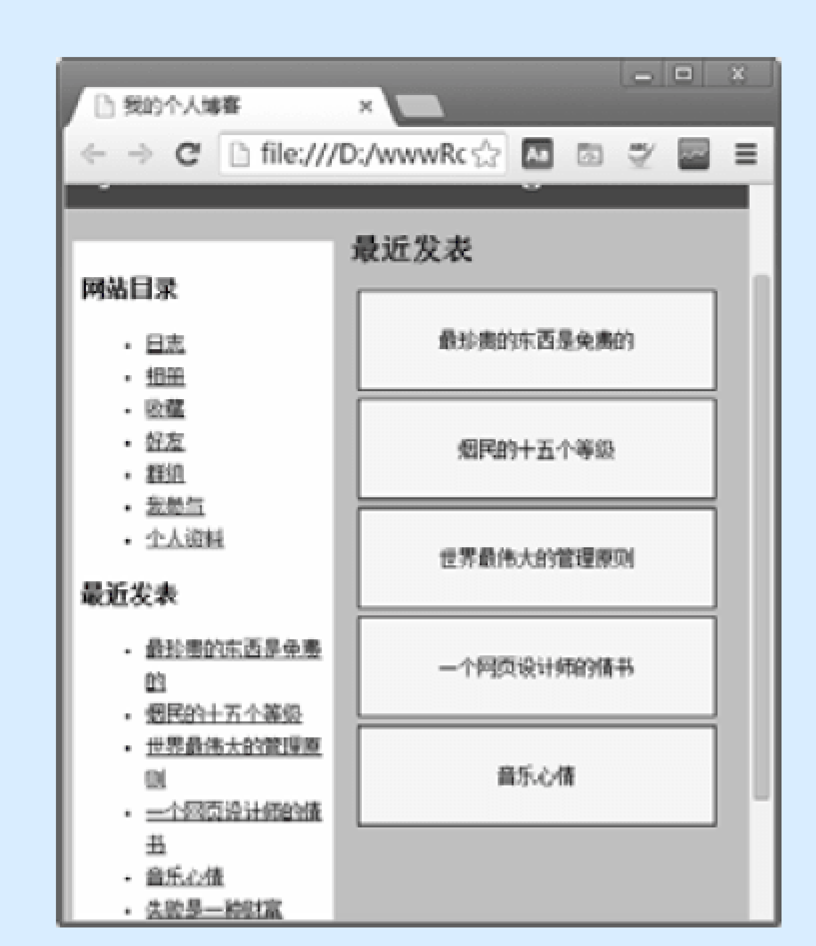


图 13-6 默认所谓垂直运行效果



图 13-7 使用box-orient属性后的水平运行效果

13.2.2 box-direction属性

box-direction 属性用于设置弹性盒模型中对象的子元素排列顺序是否反转。

其语法如下：

```
box-direction: normal | reverse | inherit
```

默认值为 normal，表示子元素按正常顺序排列。如果盒子元素的 box-orient 属性值为 horizontal，则其包含的子元素按照从左到右的顺序显示；如果盒子元素的 box-orient 属性值为 vertical，则其包含的子元素按照从上到下的顺序显示。reverse 值表示反转弹性盒模型对象的子元素的排列顺序。inherit 值表示继承上级元素的顺序。

【例 13.5】

对例 13.4 中 art 容器的样式进行修改，增加值为 reverse 的 box-direction 属性，使子元素水平显示时顺序进行反转。样式代码如下：

```
.art {
    /* 省略 box-orient 属性的设置 */
    -moz-box-direction: reverse;          /* Mozilla Gecko 引擎 */
    -webkit-box-direction: reverse;       /* Webkit 引擎 */
    box-direction: reverse;               /* 兼容标准 */
    /* 省略兼容性设置 */
}
```

重新在浏览器中运行上述代码进行测试，效果如图 13-8 所示。



图 13-8　box-direction属性的运行效果

注意：不同的浏览器中，运行出来的效果可能会有所差异。例如，Firefox 浏览器在将元素的排列顺序反转的同时，也会将元素的对齐方式进行逆转，而其他浏览器只是反转元素排列顺序。

13.2.3 box-ordinal-group属性

box-ordinal-group 属性能够设置弹性盒模型中每个子元素在盒子内的显示位置。语法如下：

```
box-ordinal-group: <integer>
```

integer 是一个从 1 开始的自然数，用来定义子元素的显示顺序。子元素的分布将根据这个属性值从小到大进行排列，即数值较小的元素显示在数值较大的元素前面。

默认情况下，子元素将根据元素的位置进行排列。如果没有指定 box-ordinal-group 属性值的子元素，则其序号默认都为 1，并且序号相同的元素将按照它们在文档中加载的顺序进行排列。另外，相同数值的元素，它们的显示顺序取决于它们的代码顺序。

【例 13.6】

创建一个示例，演示使用 box-ordinal-group 属性更改子元素显示位置前后的效果。

步骤 01　假设在一个 HTML 中有如下代码：

```
<div id="left">
    <div id="menu"></div>
    <div id="post"></div>
</div>
```

在上述代码中，ID 是 menu 的 div 元素用来显示网站的导航菜单；ID 是 post 的 div 元素用来显示最新文章标题。

步骤 02　在原来定义的样式基础上添加新的样式。定义 left 容器内的 div 元素使用盒模型，并在容器内垂直显示。主要代码如下：

```
#left{
    display: -ms-box;
    box-orient: vertical ;                    /* 兼容标准 */
    /* 省略私有属性 */
}
```

步骤 03 在浏览器中浏览页面，此时 menu 和 post 将按照页面中的定义顺序显示，如图 13-9 所示。

步骤 04 为 menu 容器使用 box-ordinal-group 属性，设置显示顺序为 2，即显示在默认元素(顺序为 1)之后。代码如下：

```
#left #menu {
    box-ordinal-group: 2;
    -webkit-box-ordinal-group: 2;
    -moz-box-ordinal-group: 2;
    -o-box-ordinal-group: 2;
}
```

步骤 05 在浏览器中重新运行页面，查看效果，如图 13-10 所示。

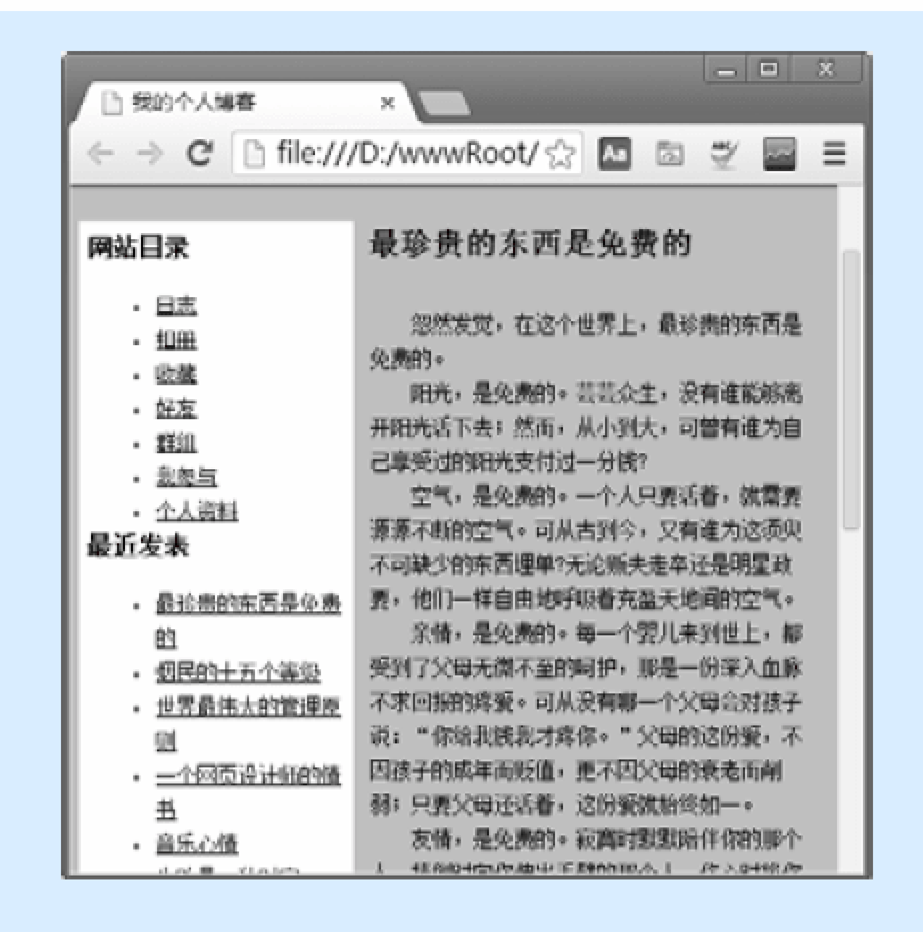


图 13-9 默认页面效果

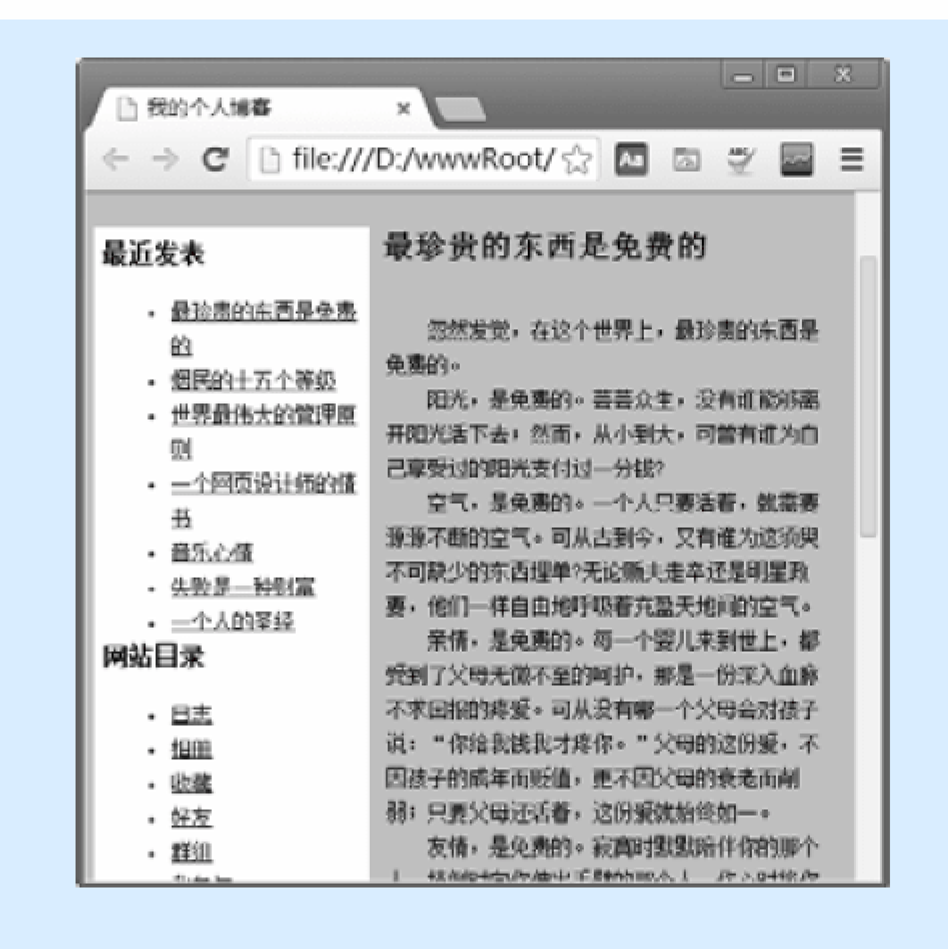


图 13-10 使用box-ordinal-group属性后的效果

13.2.4 box-flex属性

在传统的网页设计中，如果要把一个盒子分成三栏，最简单的一种方法就是把 3 个子盒子元素的宽度都设置为 33.3%。这种方法无法把父盒子的宽度完全填充，当父盒子的宽度足够大的时候，用户会看到没有填满的空白区域，这是非常不美观的。但是，如果为子元素设置了固定宽度值，弹性布局会变得更加复杂，如果使用 box-flex 属性，这个问题就会迎刃而解。

box-flex 属性解决了传统设计中习惯使用百分比定义弹性布局的弊端。其语法如下：

```
box-flex: <number>
```

number 是一个整数或者小数，初始值是 0.0。当盒子中包含多个定义了 box-flex 属性的子元素时，浏览器将会所把这些子元素的 box-flex 属性值相加，然后根据它们各自的值占总值的比例来分配盒子剩余的空间。

【例 13.7】

使用 box-flex 属性更改页面中相邻两个子元素的宽度。

步骤 01 页面中的 main 容器有两个子容器，即 middle 和 right。部分代码如下：

```
<div id="main">
  <div id="middle">  </div>
  <div id="right">  </div>
</div>
```

步骤 02 定义 main 容器的样式为 box 盒模型，并且设置子元素水平分布，具体代码这里不再给出。

步骤 03 定义 middle 容器的宽度为 5，right 容器的宽度为 4，样式代码如下：

```
#main #middle {
    -moz-box-flex: 5; /* Firefox */
    -webkit-box-flex: 5; /* Safari and Chrome */
    box-flex: 5;
    background-color: #390;
}
#main #right {
    -moz-box-flex: 4.0; /* Firefox */
    -webkit-box-flex: 4.0; /* Safari and Chrome */
    box-flex: 4;
}
```

步骤 04 在浏览器中运行上述代码，并调整浏览器的窗口大小，观察效果，效果如图 13-11 所示。



图 13-11　box-flex属性的运行效果

默认情况下，子元素并不具有弹性。只有 box-flex 属性大于 0 时，才会变得富有弹性。当子元素具有弹性时，可以通过以下方式来改变它的尺寸：

- 使用 width、height、min-width、min-height、max-width 或者 max-height 属性来定义尺寸。
- 利用盒子的尺寸来限制元素的弹性尺寸。
- 借助盒子剩余的所有空间来限制子元素的弹性尺寸。

如果子元素没有声明大小，那么其尺寸将完全取决于盒子的大小，即子元素的大小等于盒子的大小乘以它的 box-flex 属性值在所有子元素的 box-flex 属性值总和中的百分比。

用公式表示如下：

```
子元素的尺寸 = 盒子的尺寸×子元素的 box - flex 属性值/所有子元素的 box-flex 属性值的和
```

如果一个或者多个子元素中声明了具体的尺寸，那么其大小将计入其中，余下的弹性盒子将按照上面的原则分享剩下的可利用空间。

13.2.5 box-flex-group属性

box-flex-group 属性用于设置弹性盒模型中对象子元素的所属组。它动态地给数值较大的组分配其内容所需的实际内容，剩余空间则平均分给数值最小的那个组(可能有 1 个或者多个元素)。box-flex-group 属性的语法与 box-flex 属性类似，如下所示：

```
box-flex-group: <integer>
```

integer 的取值与 box-flex 属性相同，这里就不再重复介绍。

13.2.6 box-pack属性

当弹性与非弹性元素混合排版时，有可能会出现所有子元素的尺寸大于或者小于盒子的尺寸，从而出现盒子空间不足或者剩余的情况，这时就需要一种方法来管理盒子的空间。如果子元素的总尺寸小于盒子的尺寸，则可以使用 box-align 和 box-pack 属性进行管理。

box-pack 属性可以在水平方向上对盒子的剩余空间进行管理，语法如下：

```
box-pack: start | center | end | justify
```

下面对常用取值进行说明。

- start：所有子元素都在盒子的左侧，剩余的空间显示在盒子的右侧。
- center：剩余的空间在盒子的两侧平均分配。
- end：所有子元素都在盒子的右侧，剩余的空间显示在盒子的左侧。
- justify：剩余的空间在子元素之间平均分配，在第一个子元素之前和最后一个子元素之后不分配空间。

box-pack 属性受到 box-orient 属性的影响，默认情况下(即 box-orient 属性的值设置为 horizontal 时)start 和 end 所呈现的效果等同于左对齐和右对齐。当 box-orient 属性的值设置为 vertical 时，start 和 end 所呈现的效果等同于顶部对齐和底部对齐。

【例 13.8】

使用 box-pack 属性实现子元素的水平和垂直排列显示。

步骤 01 创建一个 HTML 页面。在页面中创建一个 ID 是 art 的 div 元素作为父容器，向里面添加 4 个 class 是 demo 的 div 元素作为子元素。结构代码如下：

```
<div class="art">
  <div class="demo"> </div>
  <div class="demo"> </div>
  <div class="demo"> </div>
  <div class="demo"> </div>
</div>
```

步骤 02 在 demo 容器内添加一个 h1 和一个 ul 元素。ul 元素的 ID 和 class 都是 box，代码如下：

```
<div class="demo">
    <h1>横向 box-pack 属性为 start</h1>
    <ul id="box" class="box">
        <li>1</li>
        <li>2</li>
        <li>3</li>
    </ul>
</div>
```

步骤 03 添加 class 属性值是 box 的样式，使其以盒模型显示，并让其中的子元素呈现水平方向排列。主要样式如下所示：

```
.box{
    display: box;
    margin: 0;
    padding: 10px;
    background: #000;
    list-style: none;
    box-orient: horizontal;
    width: 260px;
    height: 100px;
}
```

步骤 04 为了使 box 中的子元素在水平方向上左对齐需要使用 box-pack 属性，并设置值为 start。主要样式如下所示：

```
#box {
    -webkit-box-pack: start;
    -moz-box-pack: start;
    -o-box-pack: start;
    -ms-box-pack: start;
    box-pack: start;
}
```

步骤 05 使用相同的方法将第二个 demo 设置为水平排列，并将 box-pack 属性设置为 center，使其水平居中显示。

步骤 06 为了使 ul 中的元素垂直排列，需要将 box-orient 属性设置为 vertical。代码如下：

```
.box2 {
    box-orient: vertical;
    width: 100px;
    height: 260px;
}
```

步骤 07 此时将 box-pack 属性设置为 start 可以实现垂直顶部对齐，将 box-pack 属性设置为 end 可以实现垂直底部对齐。

步骤 08 在浏览器中运行页面，会看到 4 种对齐方式的效果，如图 13-12 所示。

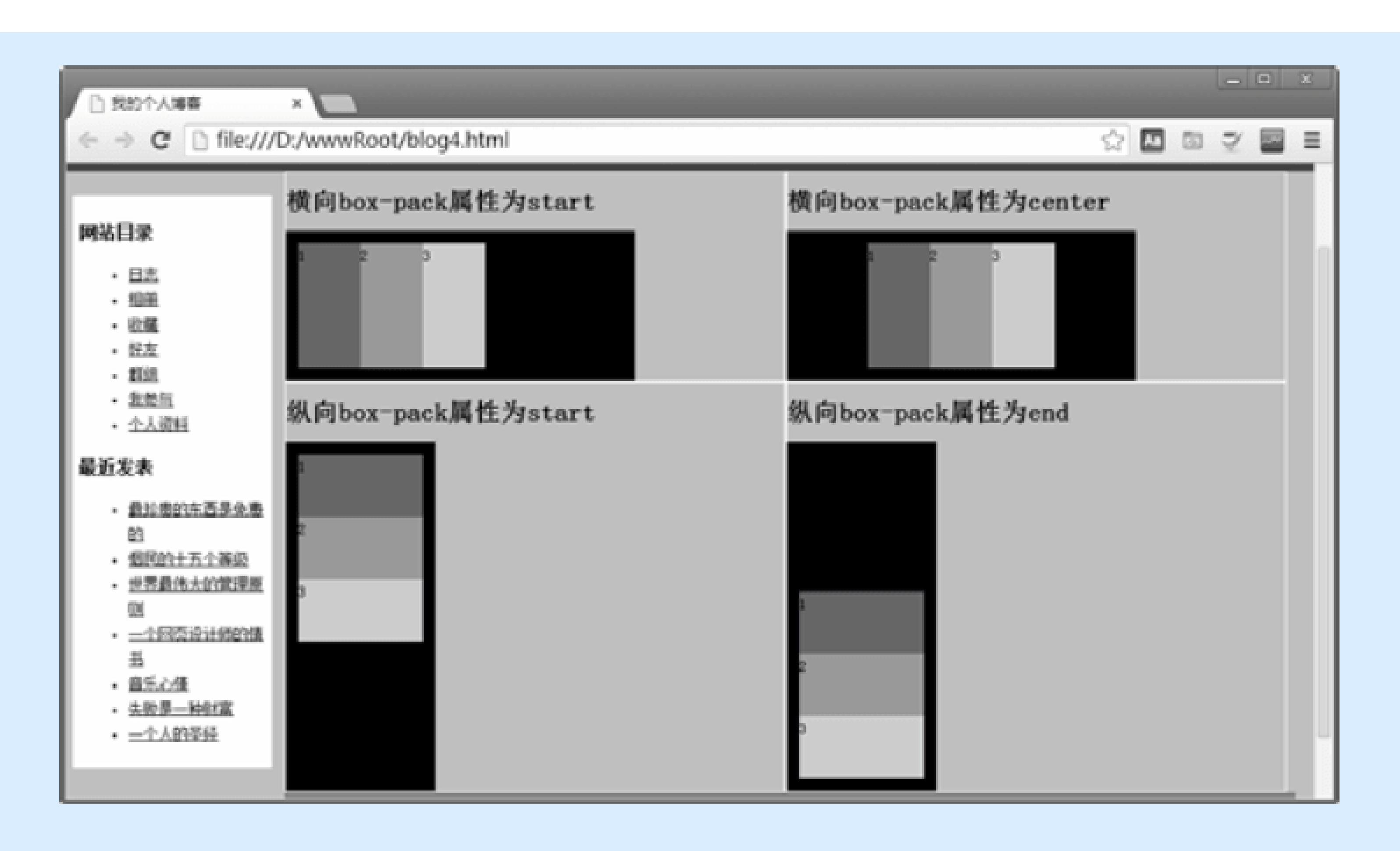


图 13-12　box-pack属性的运行效果

13.2.7　box-align属性

box-align 属性可以对盒模型中垂直方向上的剩余空间进行管理。该属性实现与 box-pack 属性相反的效果，基本语法如下：

```
box-align: start | end | center | baseline | stretch
```

下面对取值进行说明。

- start：所有子元素沿盒子的上边缘排列，都显示在盒子的上部，剩余空间显示在盒子的底部。
- end：所有子元素沿盒子的下边缘排列，都显示在盒子的底部，剩余空间显示在盒子的上部。
- center：剩余空间在盒子的上下两侧平均分配，即上面一半，下面一半。
- baseline：所有盒子沿着它们的基线排列，剩余空间可前可后显示。
- stretch：每个子元素的高度被调整到适合盒子的高度显示。

例如，将上节示例中的 box-pack 属性替换为 box-align 属性，属性值不用修改。再次在浏览器中打开，效果如图 13-13 所示。



图 13-13　box-align属性的运行效果

13.2.8 box-lines属性

CSS 3 新增了 box-lines 属性来处理弹性布局中子元素的空间溢出问题。box-lines 属性可以设置子元素是否可以换行显示。其语法如下：

```
box-lines: single | multiple
```

常用取值有两个：single 表示所有子元素都单行或者单列显示；multiple 表示所有子元素可以多行或者多列显示。

【例 13.9】

创建一个示例，演示如何在盒子内设置子元素分行显示，以避免单行显示可能会溢出盒子空间问题，实现步骤如下。

步骤 01 在页面添加一个 div 元素容器，该元素中的每一个元素都包含一张图片。部分代码如下：

```
<div id="box">
    <div><img src="images/1.png" /></div>
    <div><img src="images/2.png" /></div>
    <!-- 省略其他元素 -->
</div>
```

步骤 02 设置容器元素显示为盒型，并且定义子元素分行显示。主要样式如下：

```
#box {
    border: solid 1px red;
    width: 700px;
    height: 300px;
    display: box;
    box-lines: multiple;
    /* 省略兼容时的私有属性设置 */
}
```

步骤 03 运行 HTML 页面查看效果，Opera 浏览器中的效果如图 13-14 所示。

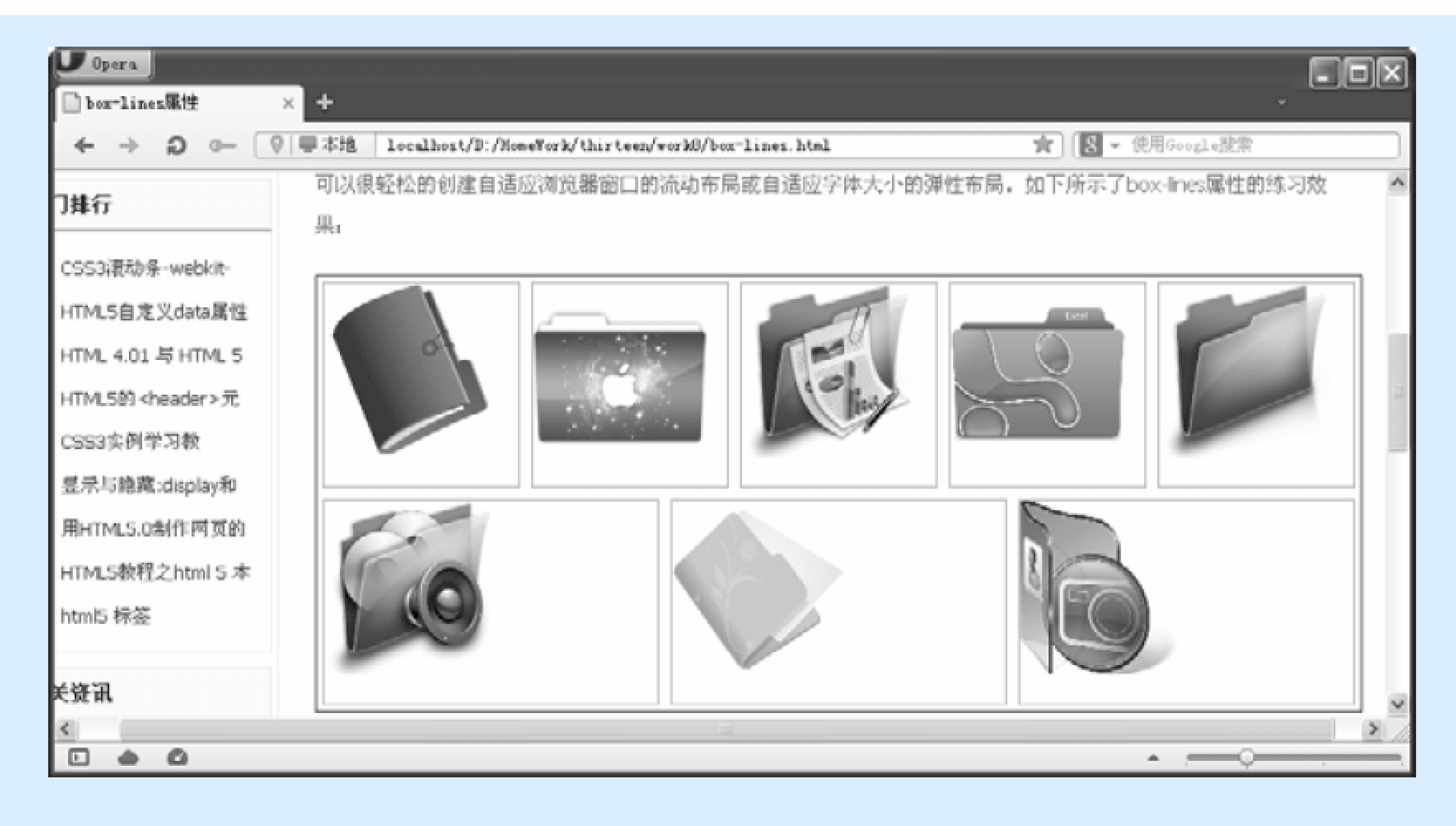


图 13-14　box-lines属性的值为multiple时的效果

注意：目前，并不是所有的主流浏览器都提供了对 box-lines 属性的支持，所以可能会导致 box-lines 属性在其他主流浏览器中显示时视为无效。另外，CSS 3 对已经存在的与盒模型有关的属性，如 overflow、overflow-x、overflow-y 和 display 属性，进行了变更，读者可以查看资料。

13.3 新增的界面布局属性

除了上面介绍的多列布局和盒模型方面的属性，CSS 3 还新增了很多与用户界面相关的属性。它们用来控制与用户界面相关效果的呈现方式，解决了用户界面的设置问题。

本节罗列 CSS 3 用户界面布局时常用的 5 个属性，分别是 box-sizing 属性、resize 属性、zoom 属性、outline-offset 属性和 nav-index 属性。

13.3.1 box-sizing属性

在网页布局中一直存在 W3C 和 IE 两种盒模型，由于这两种盒模型之间有所不同，导致浏览器的兼容性问题一直存在。为了解决这个问题，CSS 3 对盒模型进行了改进，定义了 box-sizing 属性。box-sizing 属性可以改变容器内盒模型的解析方式，其语法如下：

```
box-sizing: content-box | border-box
```

两个取值的说明如下。

(1) content-box，表示标准模式下的盒模型。对象的实际宽度的计算公式如下：

```
/*外盒尺寸计算(元素空间尺寸)*/
Element 空间高度 = content height + padding + border + margin
Element 空间宽度 = content width + padding + border + margin
/*内盒尺寸计算(元素大小)*/
Element Height = content height + padding + border (Height 为内容高度)
Element Width = content width + padding + border (Width 为内容宽度)
```

(2) border-box，表示怪异模式下的盒模型，即 IE 盒模型。对象的实际宽度的计算公式如下：

```
/*外盒尺寸计算(元素空间尺寸)*/
Element 空间高度 = content Height + margin (Height 包含了元素内容宽度、边框宽度、内距宽度)
Element 空间宽度 = content Width + margin (Width 包含了元素内容宽度、边框宽度、内距宽度)
/*内盒尺寸计算(元素大小)*/
Element Height = content Height(Height 包含了元素内容宽度、边框宽度、内距宽度)
Element Width = content Width(Width 包含了元素内容宽度、边框宽度、内距宽度)
```

假设一个元素设置宽度为 200px，内距为 10px，边框为 15px。此时，将该元素的 box-sizing 属性值设为 content-box 的效果如图 13-15 所示，将该元素 box-sizing 属性值设为 border-box 的效果如图 13-16 所示。仔细对比两个图，会看出标准盒模型与 IE 盒模型在计

算方式上的区别。

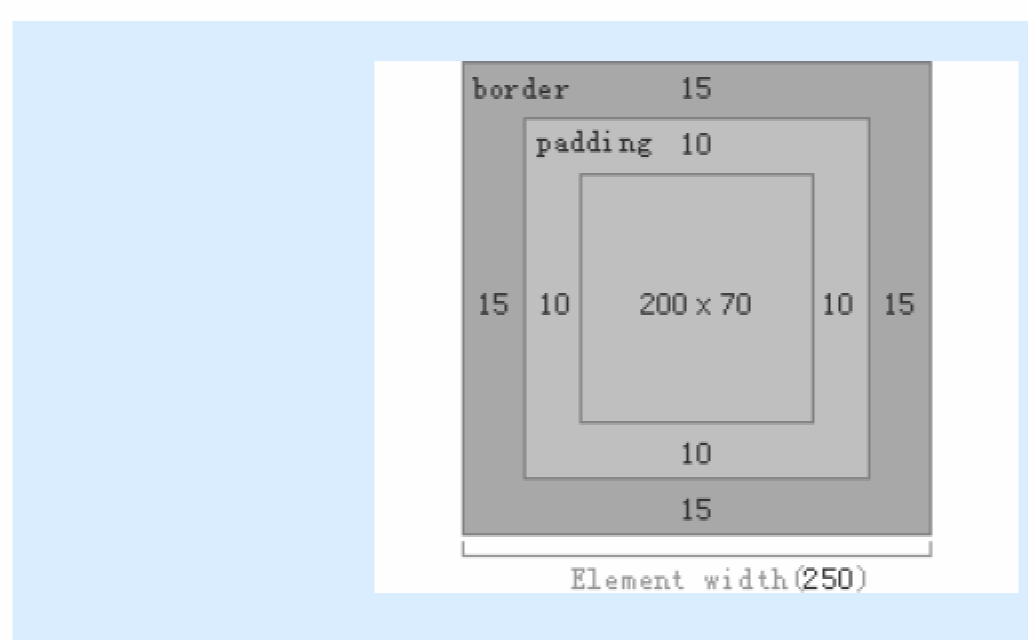


图 13-15 content-box

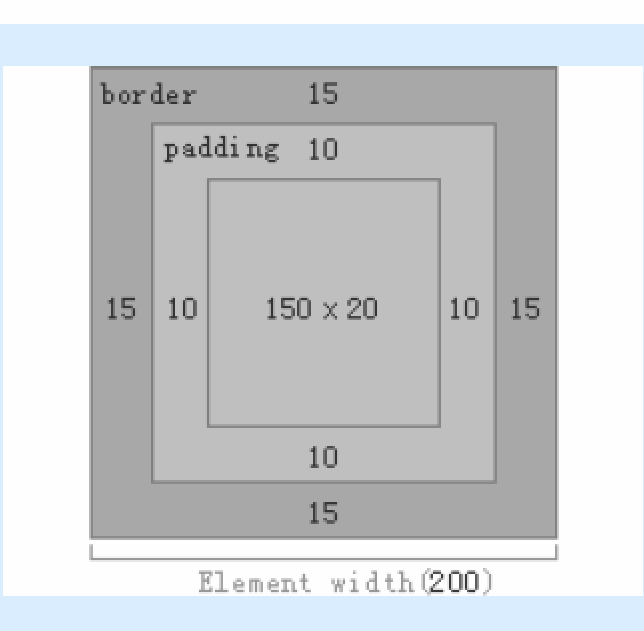


图 13-16 border-box

【例 13.10】

创建一个示例，演示元素设置 box-sizing 属性值为 content-box 和 border-box 的效果。假设有如下 HTML 页面：

```
<div class="art">
    <div><img src="images/Hydrangeas.jpg" /></div>
</div>
```

首先指定 box-sizing 属性的值为 content-box，主要样式如下：

```
.art div {
    width: 500px;                /* 宽度 */
    float: left;
    padding: 10px;               /* 填充效果 */
    border: 6px solid #999;      /* 边框效果 */
    background: #eee;            /* 背景颜色 */
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}
```

添加完成后的页面效果如图 13-17 所示。将 box-sizing 属性的值修改为 border-box，此时页面效果如图 13-18 所示。



图 13-17 值为content-box时的效果



图 13-18 值为border-box时的效果

13.3.2 resize属性

resize 属性是 CSS 3 中新增的一个非常实用的属性，它用于设置是否允许用户调整一个元素的尺寸大小。resize 属性的语法如下：

```
resize: none | both | horizontal | vertical
```

resize 属性适用于所有 overflow 属性不为 visible 的元素，各个取值说明如下。

- none：不能调整元素的尺寸。
- both：允许调整元素的宽度和高度。
- horizontal：允许调整元素的宽度。
- vertical：允许调整元素的高度。

【例 13.11】

假设在一个 HTML 页面中有如下布局代码：

```
<div>发表评论：
    <textarea name="textfield" id="textfield">评论内容</textarea>
    <input type="submit" name="button" id="button" value="提交" />
</div>
```

上述代码显示了一个用于输入评论内容的多行文本框，以及一个提交按钮。下面为该文本框添加 resize 属性，允许用户调整其宽度和高度。样式代码如下：

```
.art div textarea {
    -webkit-resize: both;
    resize: both;
    -webkit-overflow: scroll;
}
```

在浏览器中运行页面，文本框右下角会有一个可拖动的标识。如图 13-19 所示是调整尺寸前后的运行效果。



图 13-19 调整尺寸前后的运行效果

注意：使用 resize 属性时，必须定义 overflow 属性的值，否则 resize 属性的声明无效，因为元素默认的溢出显示为 visible。另外，如果想要在某个范围内自由调整宽度和高度，可以分别设置 max-width 属性和 max-height 属性，它们分别用于设置元素的最大宽度和最大高度。

13.3.3 zoom属性

zoom 属性用于设置对象的缩放比例，语法如下：

```
zoom: normal | <number> | <percentage>
```

zoom 属性适用于所有元素，取值说明如下。

- normal：表示使用对象的实际尺寸。
- number：用浮点数来定义缩放比例，它不允许负值。
- percentage：用百分比来定义缩放比例，它不允许负值。

【例 13.12】

假设在一个 HTML 页面中有如下布局代码：

```
<div class="art">
    <div><a href="#"><img src="images/Hydrangeas.jpg" width="200" /></a>
    </div>
    <div><a href="#"><img src="images/Jellyfish.jpg" width="200" /></a>
    </div>
    <div><a href="#"><img src="images/Koala.jpg" width="200" /></a></div>
    <div><a href="#"><img src="images/Lighthouse.jpg" width="200" /></a>
    </div>
    <div><a href="#"><img src="images/Penguins.jpg" width="200" /></a>
    </div>
</div>
```

下面使用 zoom 属性实现鼠标移到图片上时将图片放大 1.5 倍。样式代码如下：

```
.art a:hover{
    zoom: 150%;
}
```

在浏览器中运行 HTML 页面，查看效果，把鼠标悬浮到图片时的放大效果如图 13-20 所示。

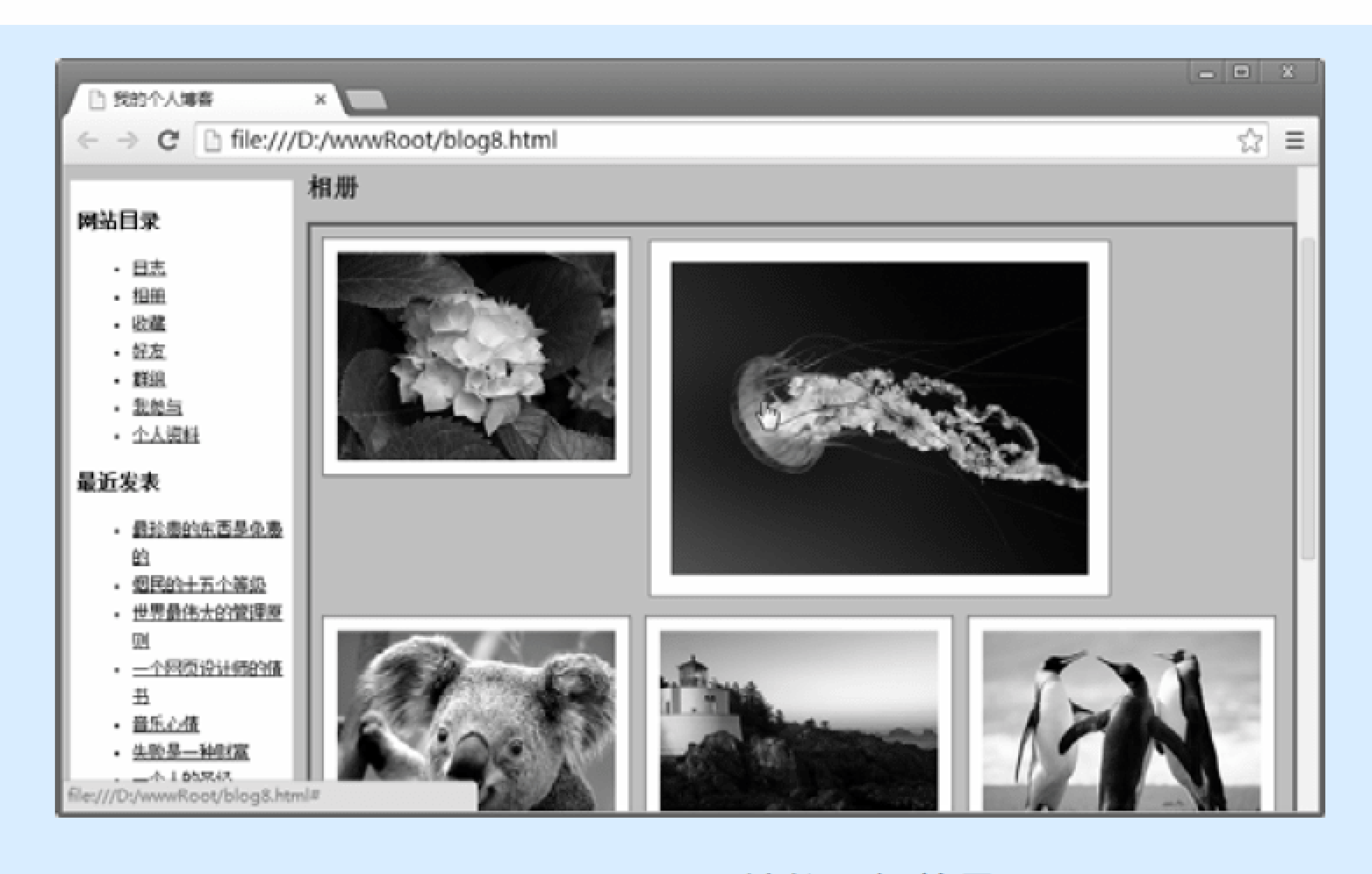


图 13-20　zoom属性的运行效果

13.3.4 outline-offset属性

在 CSS 2 中可以使用 outline 属性定义块元素的外边框线，其语法如下：

```
outline: [outline-color] | [outline-style] | [outline-width]
```

上述 3 个取值分别用于定义边框的颜色、样式和宽度，取值说明如下。

- outline-color：定义边框颜色，可接受所有的颜色。
- outline-style：定义边框的样式，与 border-style 属性的取值相同。
- outline-width：定义边框的宽度，与 border-width 属性的取值相同。

为了可以更加灵活地控制外边框线，CSS 3 中新增了 outline-offset 属性。该属性用于设置对象的外边框线偏移容器的值，适用于所有元素。语法如下：

```
outline-offset: <length>
```

outline-offset 属性的取值是由浮点数和标识符组成的长度值，它允许使用负值。取值如果为零，表示以 border 边界作为参考点，正值表示从 border 边界往外延，负值表示从 border 边界往里缩。

【例 13.13】

假设 HTML 页面中有一个留言表单用于输入姓名、联系方式、标题和留言内容。首先为页面中的文本框添加一个样式，实现在输入内容时显示一个边框线。样式代码如下：

```
input[type="text"]:focus{
    outline: #FF0 solid 5px;
}
```

在浏览器中运行页面，效果如图 13-21 所示。继续向输入框中添加样式，通过 outline-offset 属性指定获取焦点时的外边框线。样式代码如下：

```
input[type="text"]:focus{
    outline: #FF0 solid 5px;
    outline-offset: 10px;
}
```

再次运行 HTML 页面，此时的效果如图 13-22 所示。



图 13-21　outline属性的运行效果



图 13-22　outline-offset属性的运行效果

13.3.5 nav-index属性

使用 HTML 4 中的 tabindex 属性，可以控制当前文档中元素获取焦点的顺序。CSS 3 中新增加了 nav-index 属性来代替 tabindex 属性。nav-index 属性的语法如下：

```
nav-index: auto | <number>
```

auto 值表示使用浏览器默认的顺序；number 值是一个正整数，该数指定了元素的导航顺序。

默认情况下，在浏览器中使用 Tab 键可以正向按顺序导航，通过 Shift+Tab 组合键可以反向导航。因此为了使浏览器按顺序获取焦点，页面元素需要遵循以下规则：

- 该元素支持 nav-index 属性，而被赋予正整数属性值的元素将会被优先导航。
- 浏览器将按照 nav-index 值按从小到大的顺序进行导航。如果多个元素的优先级相同，则按照定义顺序进行导航。
- 对于不支持 nav-index 属性或者 nav-index 属性值为 auto 的元素，将以它们在文档流中出现的顺序进行导航。
- 对那些禁用的元素，将不参与导航。

13.4 本章习题

1. 填空题

(1) CSS 3 中设置每列固定宽度使用的属性是__________。

(2) CSS 3 中将标题横跨多行的属性是__________。

(3) __________属性用于设置弹性盒模型中子元素的排列方式。

(4) CSS 3 新增的弹性布局中__________属性用于管理空间溢出。

(5) CSS 3 中新增的__________属性可使页面上的元素允许用户调整其大小。

2. 选择题

(1) 下列不属于复合属性的是__________。

A. columns　B. box-orient　C. column-rule　D. outline-offset

(2) 下面的选择中，__________属性是 CSS 3 中新增的弹性盒模型属性。

A. overflow　B. box-direction　C. display　D. overflow-y

(3) CSS 3 中使用__________属性设置每列之间的间隔距离。

A. column-width　B. column-rule-width

C. column-gap　D. div

(4) 想要将标题横跨多列并且在中间显示，可以使用哪种办法解决？__________

A. outline-width　B. column-gap

C. column-count　D. column-span

3. 上机练习

根据本章学习的知识，使用 CSS 3 中多列布局的相关属性，设计一个以列表形式展现的相册浏览页面，参考效果如图 13-23 所示。



图 13-23　相册浏览页面





第 14 章

CSS 3 动画特效

通过前面章节的学习，相信读者已经体会到了 CSS 3 的强大。除了前面介绍的功能外，CSS 3 还可以实现元素的颜色渐变、过渡、平移以及动画等功能。在介绍 HTML 5 中新增加的绘图 canvas 元素时，曾经介绍过如何使用上下文对象绘制出渐变图形和变换(例如平移、缩放和旋转)图形。实际上，CSS 3 中新增加了一些属性，通过设置这些属性，也可以实现渐变、平移、缩放和旋转等效果。

通过本章的学习，读者不仅可以掌握新增的渐变属性，还可以掌握 CSS 3 中新增的与转换、过渡和动画有关的属性。

本章学习目标：

- 掌握线性渐变的实现
- 掌握径向渐变的实现
- 掌握 2D 转换的常用方法
- 熟悉 3D 转换的常用方法
- 掌握与过渡有关的属性
- 掌握 transition 属性的使用
- 熟悉与动画有关的属性
- 掌握@keyframes 的使用

14.1 渐 变 特 效

渐变背景是网页设计中不可或缺的审美元素，一直以来，网页设计者必须依赖现成的图片来实现渐变背景效果，这是一种笨拙的方法。CSS 3 新增加了与渐变有关的属性，通过这些属性，可以直接实现渐变效果。

基于 CSS 的渐变与图片渐变相比，最大的优点在于方便修改，同时支持无级操作，而且过渡更加自然。

14.1.1 线性渐变

线性渐变是沿着一根轴线(水平或垂直)改变颜色，从起点到终点颜色进行顺序渐变(从一边拉向另一边)。CSS 3 中使用 linear-gradient 属性实现线性渐变的功能，该属性的基本语法如下：

```
linear-gradient( [<point> || <angle>,]? <stop>, <stop> [, <stop>]* );
```

目前，CSS 渐变设计还没有一个统一的标准，用法差异很大，不同引擎实现渐变的语法不同，均为私有属性。例如，下面列出了不同引擎下实现渐变时需要添加的私有属性：

```
//Webkit 引擎
-webkit-linear-gradient([<point>||<angle>,]?<stop>,<stop>[,<stop>]*);
//Gecko 引擎
-moz-linear-gradient([<point>||<angle>,]?<stop>,<stop>[,<stop>]*);
//Presto 引擎
-o-linear-gradient([<point>||<angle>,]?<stop>,<stop>[,<stop>]*);
```

无论使用哪种渲染引擎，其使用 linear-gradient 属性的语法都是一致的。在使用 linear-gradient 属性时，需要向该属性中传入 3 个参数。其中，第一个参数表示线性渐变的方向或角度，top 是从上到下、left 是从左到右，如果定义成 left top，那就表示从左上角到右下角；第二个参数和第三个参数分别是起点颜色和终点颜色，还可以在它们之间插入多个参数，表示多种颜色的渐变。

【例 14.1】

为显示古诗的 div 元素添加背景颜色，实现颜色为红色到蓝色的渐变。步骤如下。

步骤 01 向页面中添加 div 元素，该元素包含 h1 和 p 两个元素。h1 显示古诗标题，p 显示古诗内容。页面代码如下：

```
<div class="demo" >
    <h1>蜀相</h1>
    <p>丞相祠堂何处寻，锦官城外柏森森。<br/><br/>映阶碧草自春色，隔叶黄鹂空好音。
    <br/><br/>
    三顾频烦天下计，两朝开济老臣心。<br/><br/>出师未捷身先死，长使英雄泪满襟。
    <br/><br/></p>
</div>
```

步骤 02 为上述步骤中的 div 元素添加样式，指定 div 元素的宽度、边框、高度、填

充距离、字体大小、字体颜色、字体间距以及背景等内容。代码如下：

```
.demo {
    width: 95%;
    border: 1px solid #ccc;
    height: 200px;
    padding: 10px;
    text-align: center;
    font-size: 14px;
    letter-spacing: 2px;
    color: white;
    background: -webkit-linear-gradient(top,red,blue); /*Chrome 等浏览器*/
    background: -moz-linear-gradient(top,red,blue);    /*Firefox 等浏览器*/
    background: -o-linear-gradient(top,red,blue);
}
```

步骤 03 在浏览器中运行上述代码，查看渐变效果，如图 14-1 所示。



图 14-1 两种颜色的渐变效果

步骤 04 更改上述代码，继续添加样式，通过 linear-gradient 指定多个渐变颜色，从左上角到右下角进行渐变，渐变颜色从 red 到 white 到 yellow 再到 blue，同时指定古诗字体颜色为黑色。部分样式如下：

```
.demo {
    /* 省略其他样式 */
    color: black;
    background: -webkit-linear-gradient(left top,red,white,yellow,blue);
    background: -moz-linear-gradient(left top,red,white,yellow,blue);
    background: -o-linear-gradient(left top,red,white,yellow,blue);
}
```

步骤 05 重新运行上述代码，查看渐变效果，如图 14-2 所示。

提示： 在指定颜色渐变时，除了可以直接使用颜色的英文名称外，还可以使用十六进制、RGBA 和 HSLA 等单位进行表示，感兴趣的读者可以亲自动手试一试。

在 Webkit 渲染引擎下有两种方式，上面介绍的是最新的一种发布方式，还有一种老式的语法。



图 14-2　多种颜色的渐变效果

老式语法如下：

```
-webkit-gradient(<type>, <point> [, <radius>]?, <point> [, <radius>]? [,
<stop>]*) //老式语法书写规则
```

Webkit 渲染的老式渐变是通过-webkit-gradient 实现的，它需要传入多个参数，参数说明如下。

- <type>：定义渐变的类型，linear 表示线性渐变，radial 表示径向渐变。
- <point>：定义渐变起始点和结束点坐标，即开始应用渐变的 X 轴和 Y 轴坐标，以及结束渐变的坐标。它的参数支持数值、百分比和关键字，例如(0,0)或者(left,top)等。关键字包括 top、bottom、left 和 right。
- <radius>：当定义径向渐变时，用来设置径向渐变的长度，该参数为一个数值。
- <stop>：定义渐变色和步长，包括 3 个类型值，说明如下。
    - 开始的颜色：使用 from(color value)函数进行定义。
    - 结束的颜色：使用 to(color value)函数进行定义。
    - 颜色步长：使用 color-stop(value, color value)定义。color-stop()函数包含两个参数值，第一个参数值为一个数值或者百分比值，取值范围为 0~1.0(或者 0%~100%)；第二个参数值表示任意的颜色值。

【例 14.2】

重新定义例 14.1 中的样式内容，通过-webkit-gradient 实现#9F0 ~ #FCF 的颜色渐变。代码如下：

```
.demo {
    /*省略其他样式*/

    color: white;
    background:
  -webkit-gradient(linear,center top,center bottom,from(#9F0),to(#FCF));
}
```

运行本例的代码，查看渐变效果，如图 14-3 所示。



图 14-3 #9F0 ~ #FCF的颜色渐变

可以直接定义步长，而不通过 from()和 to()进行指定。代码如下：

```
.demo {
    /*省略其他样式*/
    color: white;
    background: -webkit-gradient(linear,center top,center bottom,
      color-stop(0.3,blue), color-stop (0.7,orange));
}
```

重新运行上述代码查看渐变效果，如图 14-4 所示。



图 14-4 color-stop()的使用

还可以将 from()、to()与 color-stop()结合起来使用，实现多重渐变。样式代码如下：

```
.demo {
    /*省略其他样式*/
    color: white;
    background: -webkit-gradient(linear,center top,
     center bottom,from(#9F0),to(#FCF), color-stop (0.3,blue),
     color-stop(0.7,orange));
}
```

继续运行上述代码，查看渐变效果，如图 14-5 所示。

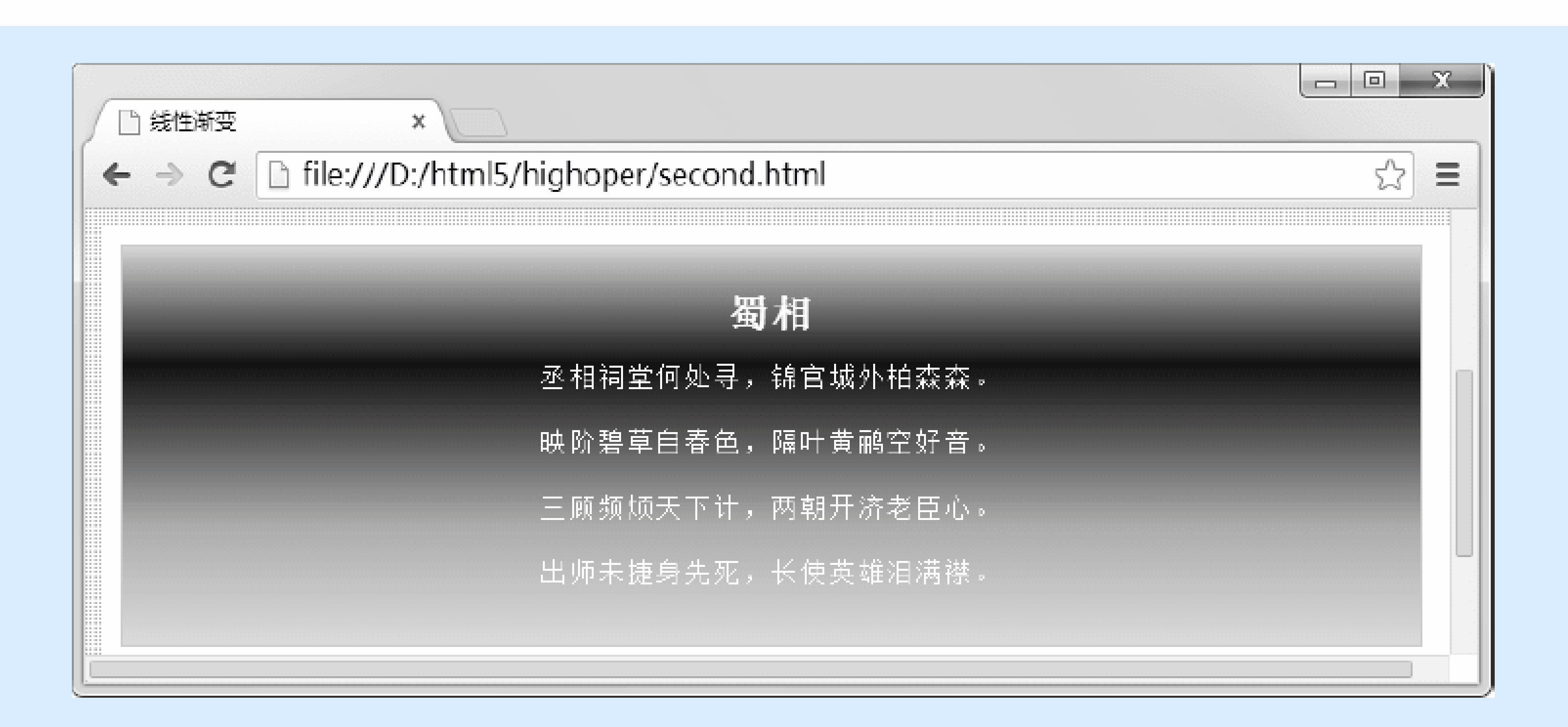


图 14-5　多重渐变的实现

14.1.2　径向渐变

径向渐变是指从起点到终点、颜色从内到外进行圆形渐变(从中间向外拉)。CSS 3 新增 radial-gradient 实现径向渐变，基本语法如下：

```
-webkit-radial-gradient([<point> || <angle>,]? [<shape> || <size>,]?
  <stop>, <stop>[, <stop>]*)
-moz-radial-gradient([<point> || <angle>,]? [<shape> || <size>,]?
  <stop>, <stop>[, <stop>]*)
-o-radial-gradient([<point> || <angle>,]? [<shape> || <size>,]?
  <stop>, <stop>[, <stop>]*)
```

上述语法中，point 表示定义渐变的起始点；angle 定义渐变的角度；shape 定义渐变的形状，它的值包含 circle(圆)和 ellipse(椭圆，默认值)；size 定义圆半径，或者椭圆的轴长度，通过 closest-side、closest-corner、farthest-side、farthest-corner、contain 和 cover 等关键字定义；stop 表示可以调用 color-stop()函数定义步长。

【例 14.3】

本例继续利用前面的页面，为古诗背景添加径向渐变的效果。实现步骤如下。

步骤 01 向页面中添加 div 元素，该元素包含 h1 和 p 两个元素，页面代码可以参考例 14.1 中的步骤 01。

步骤 02 添加简单的径向渐变，渐变的开关为圆形，渐变的形式是从黄色到蓝色。主要样式如下：

```
.demo {
    /*省略其他样式*/
    color: white;
    background: -webkit-radial-gradient(circle,yellow,blue);
}
```

步骤 03 在浏览器中运行上述页面，查看效果，如图 14-6 所示。

步骤 04 更改上述样式代码，再次实现径向渐变，形状为椭圆。从中间向外由红色、绿色到蓝色渐变显示，并设置渐变尺寸为 cover 关键字。代码如下：

```
.demo {
    /*省略其他样式*/
```

```
    color: white;
    background: -webkit-radial-gradient(ellipse cover,red,green,blue);
}
```



图 14-6　径向渐变效果 1

步骤 05　重新运行上述代码，查看效果，如图 14-7 所示。



图 14-7　径向渐变效果 2

在 Webkit 渲染引擎下，可以直接通过-webkit-gradient 实现径向渐变，在实现径向渐变时，需要指定两个圆，包括圆心坐标和半径长度。

【例 14.4】

下面使用-webkit-gradient 实现一个径向渐变，圆心坐标为(200,100)，内圆半径为 10，外圆坐标为 100，内圆小于外圆半径，从内圆浅蓝色到外圆绿色径向渐变，超出外圆半径显示为绿色，内圆显示为浅蓝色。样式如下：

```
.demo {
    /*省略其他样式*/
    color: black;
    background: -webkit-gradient(radial,200 100,10,200 100,100,
      from(lightblue),to(green));
}
```

上述代码在实现径向渐变的效果时，径向渐变的起点坐标(200,100)和结束坐标(200,100)分别定义内圆和外圆坐标，这里指定的坐标相同，是一个同心圆。定义坐标时还指定了内圆和外圆的半径，from 和 to 分别指定渐变开始和结束时的颜色。在浏览器中运

行上述代码，查看径向渐变效果，如图 14-8 所示。



图 14-8　-webkit-gradient实现径向效果

提示：在实现渐变时，有一种特殊的渐变——重复渐变，重复渐变就是将渐变的效果在元素中重复显示。读者在实现重复线性渐变和重复径向渐变的效果时，需要分别使用 repeating-linear-gradient 属性和 repeating-radial-gradient 属性。读者可以亲自动手练习使用这两个属性，这里不再详细说明。

14.2　转　　换

转换是使元素改变形状、尺寸和位置的一种效果。通过 CSS 3 转换，Web 设计者能够对元素进行移动、缩放、转换、拉长或者拉伸。

14.2.1　2D转换

Web 开发者可以通过 2D 或者 3D 来转换元素，CSS 3 中新增的转换属性包含 transform 和 transform-origin 两个。transform 属性向元素应用 2D 或者 3D 转换；transform-origin 属性允许改变被转换元素的位置。除了这两个属性外，还提供了多个方法，表 14-1 列出了一些 2D 转换方法。

表 14-1　2D 转换方法

方法名称	说　明
matrix(n,n,n,n,n,n)	定义 2D 转换，使用 6 个值的矩阵
translate(x,y)	定义 2D 转换，沿着 X 轴和 Y 轴移动元素
translateX(n)	定义 2D 转换，沿着 X 轴移动元素
translateY(n)	定义 2D 转换，沿着 Y 轴移动元素
scale(x,y)	定义 2D 缩放转换，改变元素的宽度和高度
scaleX(n)	定义 2D 缩放转换，改变元素的宽度

续表

方法名称	说　明
scaleY(n)	定义 2D 缩放转换，改变元素的高度
rotate(angle)	定义 2D 旋转，在参数中规定角度
skew(x-angle,y-angle)	定义 2D 倾斜转换，沿着 X 轴和 Y 轴
skewX(angle)	定义 2D 倾斜转换，沿着 X 轴
skewY(angle)	定义 2D 倾斜转换，沿着 Y 轴

1. translate()方法

translate()、translateX()和 translateY()方法都可以实现平移的效果，通过 translate()方法，元素可以从当前位置移动到另一位置。在移动时，需要根据指定的 X 坐标和 Y 坐标进行移动，移动效果如图 14-9 所示。在该图中，实线表示平移前的元素，虚线表示平移后的元素。

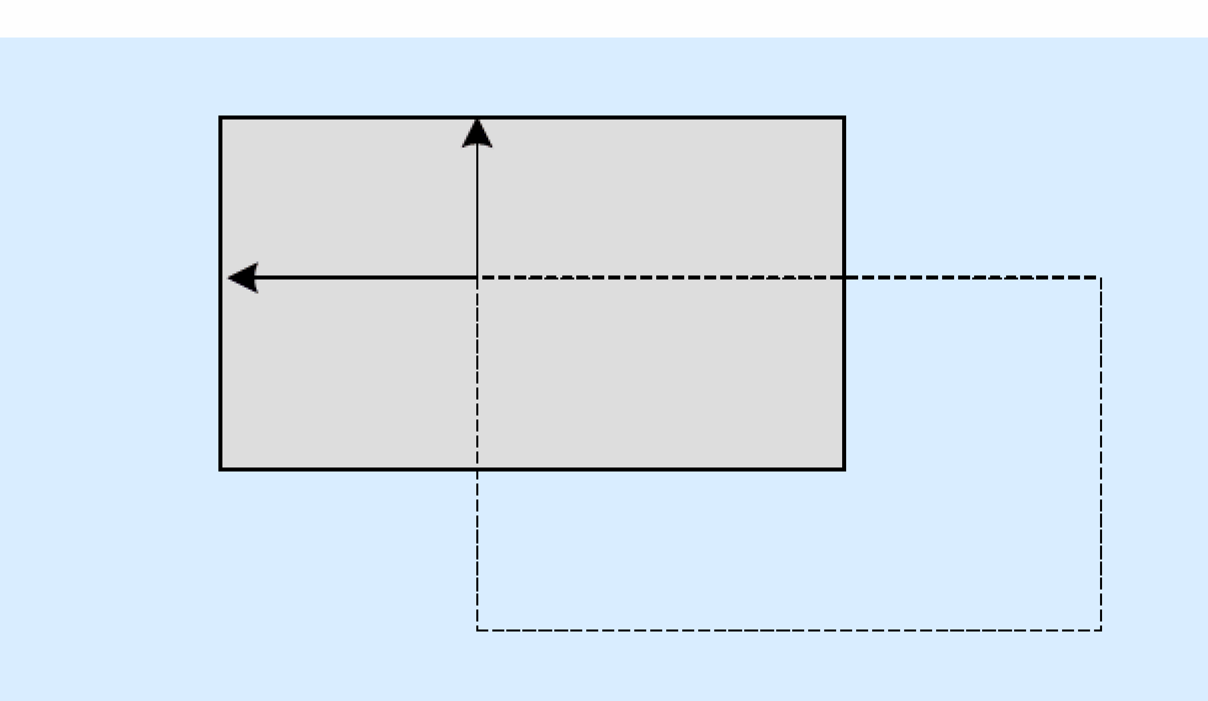


图 14-9　translate()方法平移的效果

【例 14.5】

在先前的示例中，显示古诗信息时，通过 text-align 属性设置文本内容居中显示。现在，为了演示 translate()方法的实现效果，通过平移将古诗内容居中。实现步骤如下。

步骤 01 向页面中添加 div 元素，该元素包含 h1 元素和 p 元素。h1 元素显示古诗标题，p 元素显示古诗内容。

步骤 02 为 div 元素指定宽度、高度、字体颜色和渐变背景等属性。部分代码如下所示：

```
.demo {
    width: 95%;
    border: 1px solid #ccc;
    height: 200px;
    color: black;
    background: -webkit-linear-gradient(top,#FFB,#A6FFA6); /*Chrome 等浏览器*/
    /*省略其他样式*/
}
```

步骤 03 在浏览器中运行页面，查看默认情况下的显示效果，如图 14-10 所示。

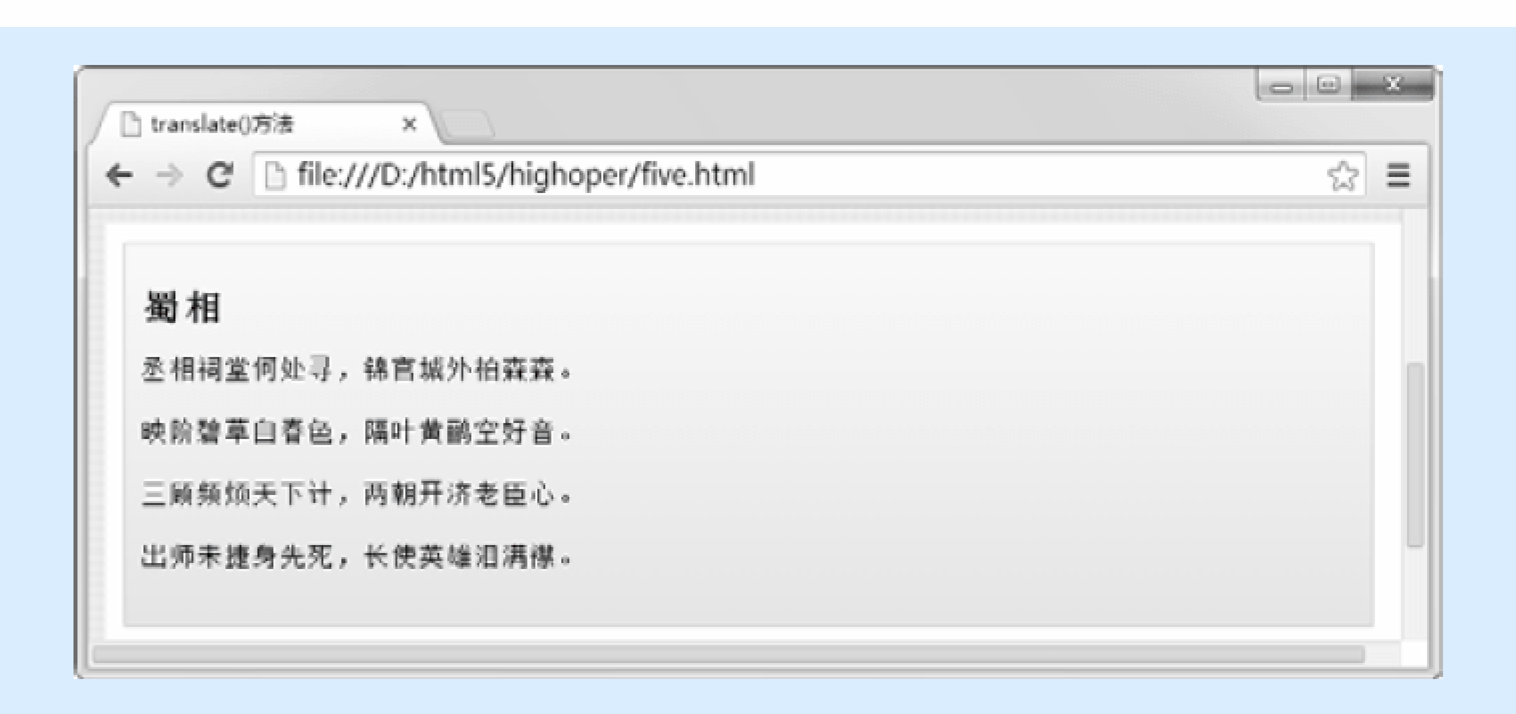


图 14-10　页面初始效果

步骤 04　为页面中的 h1 元素和 p 元素设计样式，通过 translate()方法实现平移效果。代码如下：

```
.demo h1{
    transform: translate(300px,0px);
    -moz-transform: translate(300px,0px);
    -webkit-transform: translate(300px,0px);
    -o-transform: translate(300px,0px);
}
.demo p{
    transform: translate(200px,0px);
    -moz-transform: translate(200px,0px);
    -webkit-transform: translate(200px,0px);
    -o-transform: translate(200px,0px);
}
```

步骤 05　刷新页面，查看平移效果，如图 14-11 所示。



图 14-11　平移后的效果

2. rotate()方法

通过 rotate()方法，元素顺时针旋转给定的角度。该方法中的参数允许传入负值，传入负值时元素将逆时针旋转。

【例 14.6】

继续利用前面页面中的古诗页面进行更改，指定古诗内容逆时针旋转 5 度。以 Webkit 渲染引擎为例。样式代码如下：

```
.demo p{
    -webkit-transform: rotate(-5deg);
}
```

重新运行本次示例的代码，观察旋转效果，如图 14-12 所示。



图 14-12　旋转效果

3. scale()方法

通过 scale()方法，元素的尺寸会增加或者减少。使用 scale()时需要传入两个参数，根据给定的宽度(X 轴)和高度(Y 轴)实现缩放。

【例 14.7】

本例通过 scale()方法实现图片的缩放，用户可以根据网页中的滑块查看缩放效果。实现步骤如下。

步骤 01　向页面中添加两个 range 类型的 input 元素、一个 button 按钮、一个 span 元素以及一个 img 元素。其中，range 类型的 input 元素显示滑块效果，button 按钮执行操作，span 元素显示当前 X 轴和 Y 轴的缩放值，img 元素显示图片。具体代码如下：

```
X 轴：<input id="myx" type="range" min="0.5" max="3" step="0.1" value="1"
onChange="ChangeScale()" />
Y 轴：<input id="myy" type="range" min="0.5" max="3" step="0.1" value="1"
onChange="ChangeScale()" />
<input id="myb" type="button" value="恢复正常" onClick="ChangeNormal()" />
<span id="show"></span><br/><br/>
<img id="myimg" src="images/first.jpg" width="300" height="200">
```

步骤 02　从上个步骤中可以看出，为滑块和按钮分别指定了 onChange 和 onClick 两个事件属性。当改变滑块的值时会调用 ChangeScale()函数获取滑块的值，并且将值显示到页面，同时对图片进行缩放。ChangeScale()函数的代码如下：

```
function ChangeScale(){
    var x = document.getElementById("myx").value;          //X 轴缩放值
    var y = document.getElementById("myy").value;          //Y 轴缩放值
    var img = document.getElementById("myimg");            //img 对象
    img.style.webkitTransform = "scale("+x+","+y+")";      //实现缩放效果
    document.getElementById("show").innerHTML = "X 轴："+x+",Y 轴："+y;
}
```

步骤 03　单击按钮时会触发 Click 事件调用 ChangeNormal()函数，该函数将恢复图片的原始尺寸。代码如下：

```
function ChangeNormal(){
    var img = document.getElementById("myimg");
    img.style.webkitTransform = "scale(1,1)";
}
```

步骤 04　在浏览器中运行上述代码，查看效果，初始效果和单击"恢复正常"按钮时的效果一致，如图 14-13 所示。

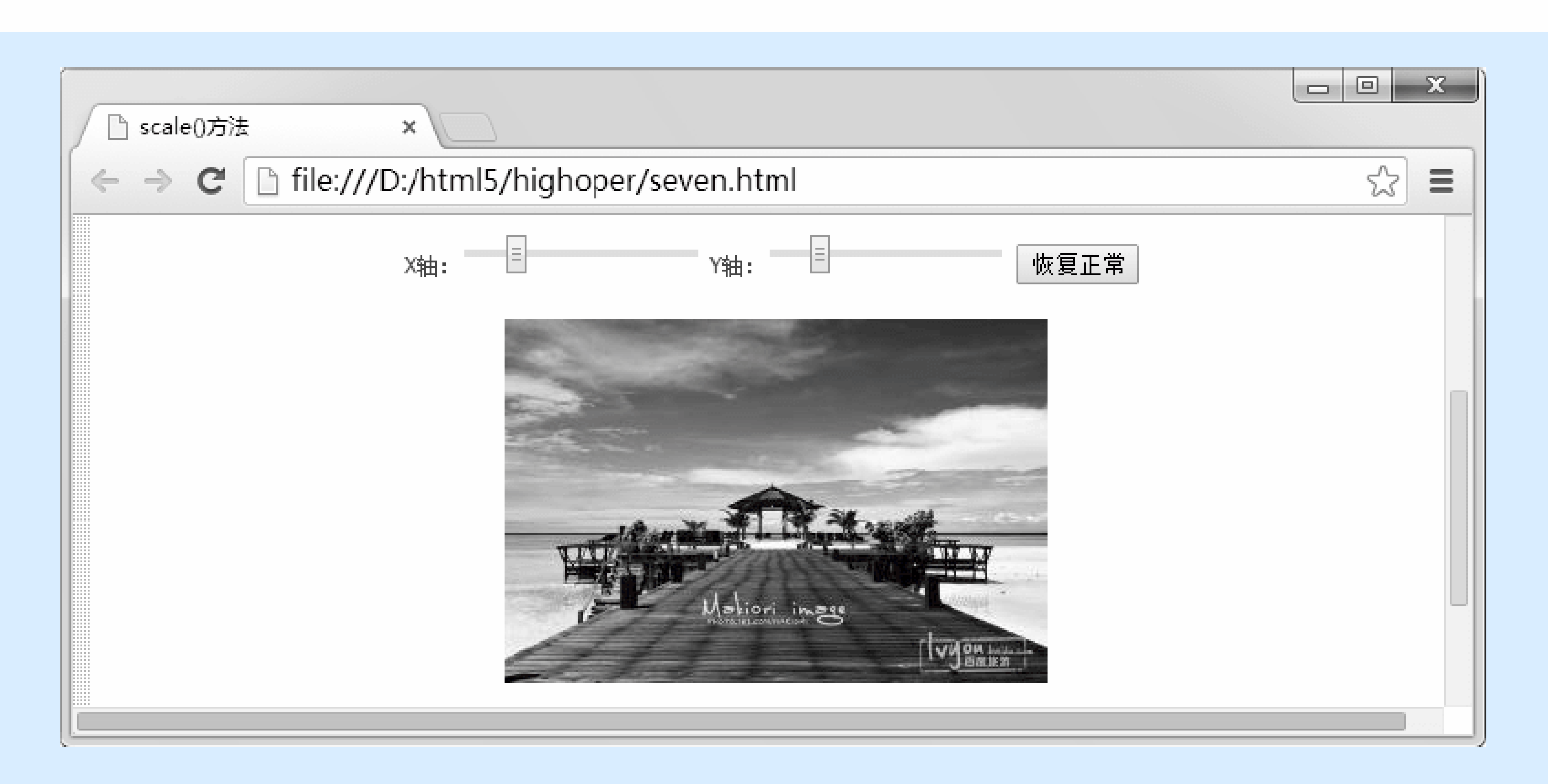


图 14-13　初始效果

步骤 05　拖动图 14-13 中的滑块对图片进行缩放，X 轴扩大到原宽度的 1.4 倍，Y 轴为原高度的一半，效果如图 14-14 所示。

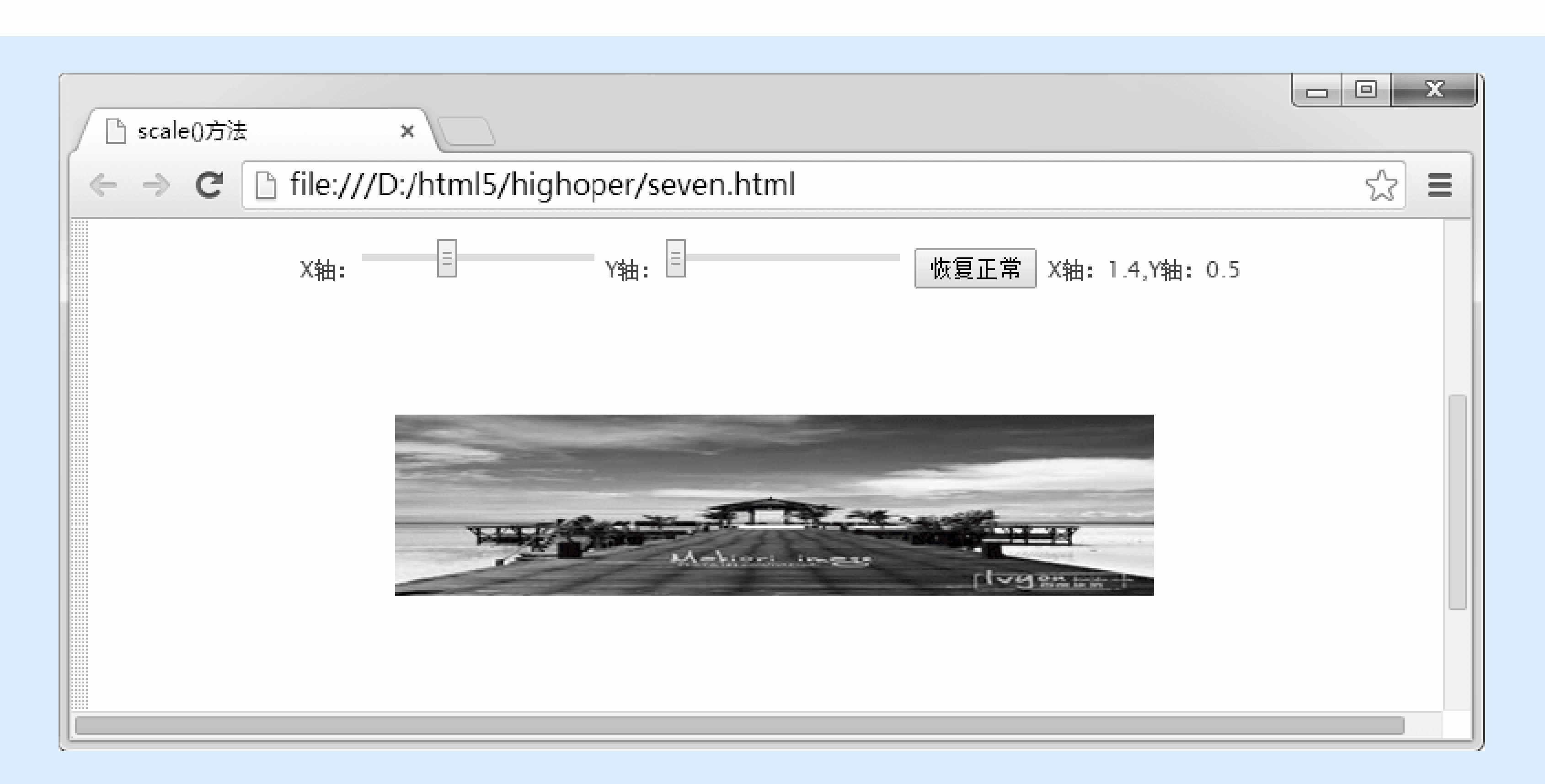


图 14-14　缩放效果

提示：通过 JavaScript 更改变换或者过渡等效果时，由于渲染引擎不同，因此需要添加不同的私有属性。该示例以 Webkit 引擎为例，将样式中的-webkit-transform 更改为 webkitTransform 属性。如果浏览器使用 Gecko 引擎，那么 JavaScript 中应该使用 mozTransform 属性。

4. skew()方法

通过 skew()方法，元素翻转给定的角度。使用时需要传入两个参数，这两个参数分别表示沿着水平和垂直方向倾斜。其中，第二个参数可以省略，省略该参数时，默认值为 0。

【例 14.8】

直接向页面中添加一张图片，页面显示时实现翻转功能。通过 skew(20deg,10deg)指定围绕 X 轴把元素翻转 20 度，围绕 Y 轴翻转 10 度。样式代码如下：

```
#myimg{
    transform: skew(30deg,20deg);
    -webkit-transform: skew(20deg,10deg);
    /* 省略其他私有属性的设置 */
}
```

在浏览器中访问本示例的页面，查看翻转效果，如图 14-15 所示。

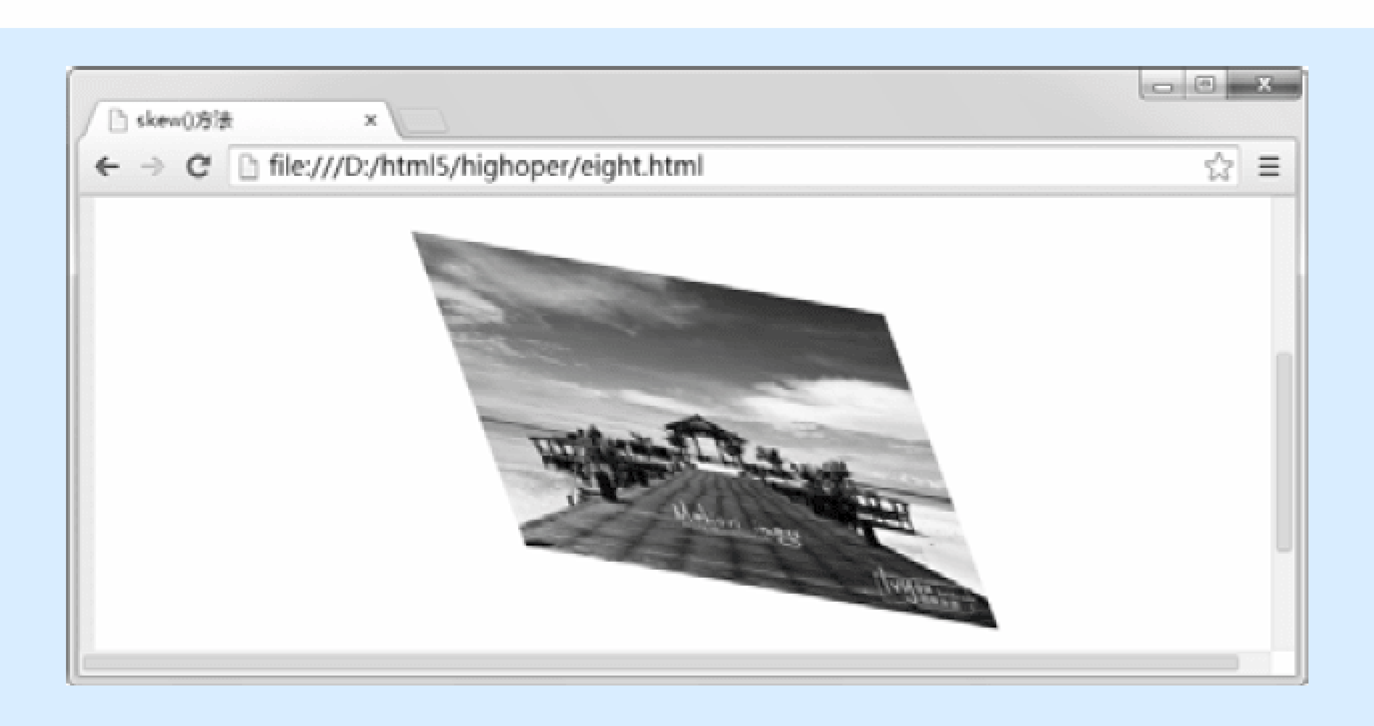


图 14-15　图片的翻转效果

14.2.2　3D转换

3D 是 3 Dimensions 的简称，中文可称为三维、三个维度或者三个坐标，即有长、宽、高。CSS 3 中新增了实现 3D 转换的属性和方法，表 14-2 和表 14-3 分别对这些属性和方法进行说明。

表 14-2　3D 转换的属性

属性名称	说　明
transform	向元素应用 2D 或 3D 转换
transform-origin	允许改变被转换元素的位置
transform-style	指定被嵌套元素如何在 3D 空间中显示。其值包括 flat 和 preserve-3d
perspective	指定 3D 元素的透视效果
perspective-origin	指定 3D 元素的底部位置
backface-visibility	定义元素在不面对屏幕时是否可见

表 14-3　3D 转换的方法

方法名称	说　明
matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	定义 3D 转换，使用 16 个值的 4×4 矩阵
translate3d(x,y,z)	定义 3D 转换
translateX(x)	定义 3D 转换，仅使用用于 X 轴的值
translateY(y)	定义 3D 转换，仅使用用于 Y 轴的值
translateZ(z)	定义 3D 转换，仅使用用于 Z 轴的值
scale3d(x,y,z)	定义 3D 缩放转换
scaleX(x)	定义 3D 缩放转换，通过给定一个 X 轴的值
scaleY(y)	定义 3D 缩放转换，通过给定一个 Y 轴的值
scaleZ(z)	定义 3D 缩放转换，通过给定一个 Z 轴的值
rotate3d(x,y,z,angle)	定义 3D 旋转
rotateX(angle)	定义沿 X 轴的 3D 旋转
rotateY(angle)	定义沿 Y 轴的 3D 旋转
rotateZ(angle)	定义沿 Z 轴的 3D 旋转
perspective(n)	定义 3D 转换元素的透视视图

【例 14.9】

表 14-2 和表 14-3 列出了一系列 CSS 3 中支持 3D 转换的属性和方法，本例介绍如何使用 rotateX()、rotateY()和 rotateZ()方法实现 3D 旋转效果。实现步骤如下。

步骤 01 向页面中添加 3 个滑块、一个 span 元素和一张图片。页面代码如下：

```
X 轴：<input id="myx" type="range" min="1" max="360" step="1"
  onChange="ChangeRotate()" />
Y 轴：<input id="myy" type="range" min="1" max="360" step="1"
  onChange="ChangeRotate()" />
Z 轴：<input id="myz" type="range" min="1" max="360" step="1"
  onChange="ChangeRotate()" />
<span id="show"></span><br/><br/>
<img id="myimg" src="images/first.jpg" width="300" height="200"
  style="padding-top:50px;">
```

步骤 02 在上个步骤中，为每一个 range 类型的 input 元素添加了 Change 事件。改变滑块时会调用 ChangeRotate()函数达到旋转效果。函数代码如下：

```
function ChangeRotate(){
    var x = document.getElementById("myx").value;
    var y = document.getElementById("myy").value;
    var z = document.getElementById("myz").value;
    var show = document.getElementById("show");
    show.innerHTML = "X 轴旋转"+x+"度，Y 轴旋转"+y+"度，Z 轴旋转"+z+"度";
    var img = document.getElementById("myimg");
    img.style.webkitTransform = "rotateX("+x+"deg)";
    img.style.webkitTransform += "rotateY("+y+"deg)";
    img.style.webkitTransform += "rotateZ("+z+"deg)";
}
```

上述代码首先获取滑块沿着 X 轴、Y 轴和 Z 轴旋转的角度，接着将旋转角度显示到页面，然后再获取页面中的图片对象，最后通过 webkitTransform 属性设置旋转效果。

步骤 03 页面加载时自动调用 ChangeRotate()函数实现旋转，代码如下：

```
window.onload = ChangeRotate();
```

步骤 04 在浏览器中运行上述代码，查看效果，如图 14-16 所示。



图 14-16 初始效果

步骤 05 分别拖动图 14-16 中的滑块实现 X 轴、Y 轴和 Z 轴的旋转，X 轴旋转 120 度、Y 轴旋转 50 度、Z 轴旋转 38 度时的效果如图 14-17 所示。



图 14-17 图片旋转效果

提示： 本节介绍了多个 3D 转换的属性和方法，上述示例演示 rotateX()、rotateY()和 rotateZ()方法的使用。读者还可以使用表 14-2 和表 14-3 中的属性和方法进行练习，这里不再动手实现。

14.3 过　渡

CSS 3 中新增加了过渡属性，使用这些属性可以在不使用 Flash 动画或者 JavaScript 的情况下，当元素从一种样式变换为另一种样式时为元素添加效果。简单地说，CSS 3 过渡是指元素从一种样式逐渐改变为另一种的效果。要实现这一点，必须指定两项内容：指定希望把效果添加到哪个 CSS 属性上；指定效果的时长。

14.3.1 常用的单个属性

CSS 3 中新增加了多个属性，本节首先介绍实现过渡时常用的 4 个属性：transition-property、transition-duration、transition-timing-function 和 transition-delay 属性。

1. transition-property属性

transition-property 属性指定应用过渡效果的 CSS 属性的名称，过渡效果通常在用户将鼠标指针浮动到元素上时发生。当指定的 CSS 属性改变时，过渡效果将开始。transition-property 属性的基本语法如下：

```
transition-property: none | all | property;
```

从上述语法可以看出：transition-property 属性的可能取值包括 none、all 和 property，说明如下。

- none：没有属性会获得过渡效果。
- all：所有属性都将获得过渡效果。
- property：定义应用过渡效果的 CSS 属性名称列表，列表以逗号分隔。

提示：在脚本中设置 transition-property 属性时使用的代码是 object.style.transitionProperty = “width,height”。使用该属性或下面介绍的属性时也需要考虑到浏览器的渲染引擎，例如对 Chrome 浏览器来说，需要添加 Webkit 私有属性，即脚本实现代码是 object.style.WebkitTranstionProperty=“width,height”。

2. transition-duration属性

在使用 transition-property 属性时，必须设置 transition-duration 属性，否则时长为 0，就不会产生任何过渡效果。transition-duration 属性用于指定过渡经过的时间，即指定从旧样式属性换到新样式需要多长时间才能完成，默认单位是秒或者毫秒。如果将该属性的值设置为非负数或者不设置，则会被视为 0。该属性的基本语法如下：

```
transition-duration: time;
```

【例 14.10】

本例通过设置 transition-property 和 transition-duration 属性在 2 秒内实现 div 元素宽度和高度从 100 到 200 的过渡。步骤如下。

步骤 01　向页面中添加 div 元素，这里省略页面代码。

步骤 02　为 div 元素指定样式，设置该元素的宽度、高度、背景颜色以及过渡名称和过渡时间。主要代码如下：

```
div{
    width: 100px;
    height: 100px;
    background: -webkit-linear-gradient(left top,blue,green);
    -webkit-transition-property: width,height;      /* 过渡名称是 width */
    -webkit-transition-duration: 2s;                /* 过渡时间为 2 秒 */
    /*省略其他浏览器的私有属性设置*/
}
```

步骤 03　为 div 元素指定悬浮时的效果，悬浮时该元素的宽度和高度都为 200 像素，这里省略悬浮时的样式代码。

步骤 04　在浏览器中运行上述代码进行测试，将鼠标指针悬浮到 div 元素，查看效果，过渡时的效果如图 14-18 所示。

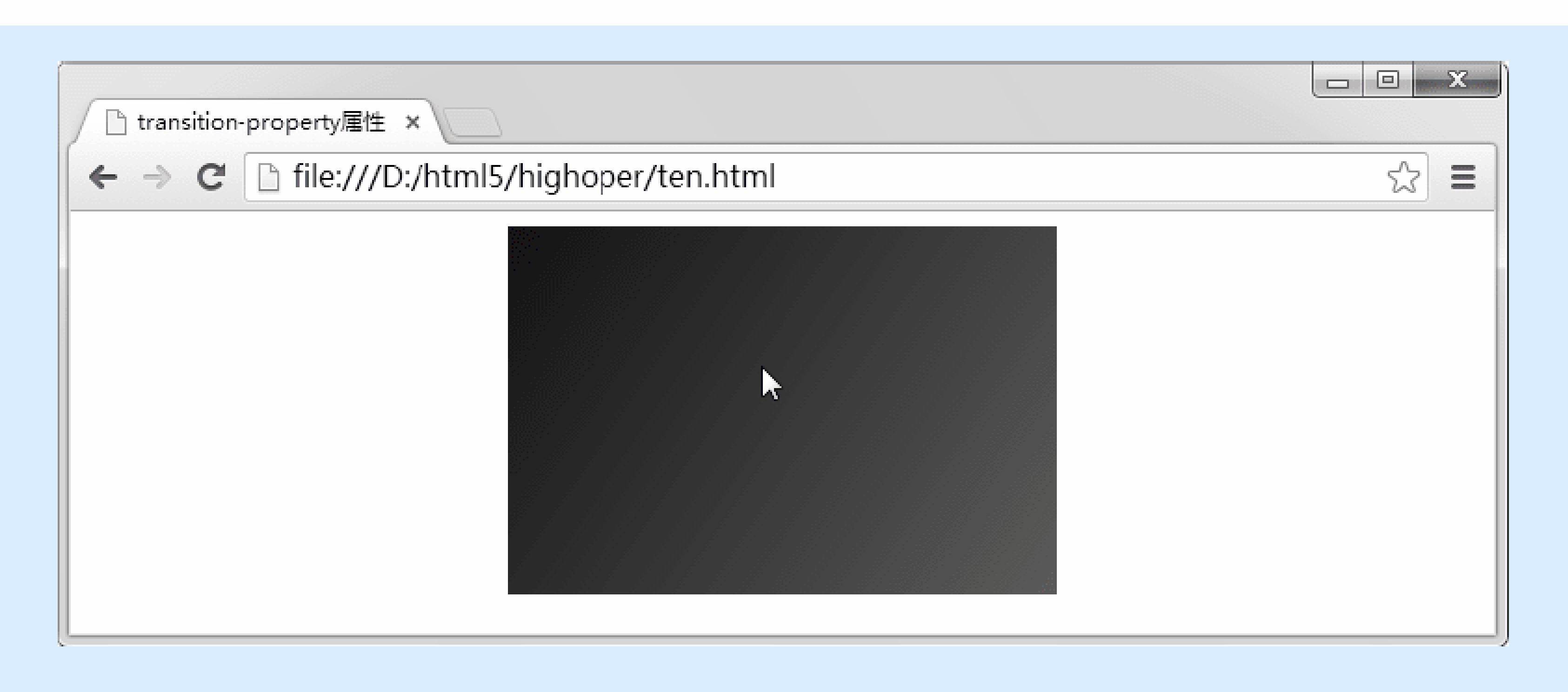


图 14-18　宽度和高度过渡时的效果

3. transition-timing-function属性

transition-timing-function 属性指定过渡效果的时间曲线。基本语法如下：

```
transition-timing-function: linear | ease | ease-in | ease-out
  | ease-in-out | cubic-bezier(n,n,n,n);
```

从上述语法中可以看出，transition-timing-function 属性的取值有 6 个，说明如下。

- linear：默认值，指定切换效果以相同速度从开始到结束。将属性值设置为 linear 的效果等同于 cubic-bezier(0.0,0.0,1.0,1.0)。
- ease：指定一个缓慢的开始，然后加快，最后慢慢结束。属性值设置为 ease 的效果等同于 cubic-bezier(0.25,0.1,0.25,1.0)。
- ease-in：指定一个缓慢的开始，然后逐渐加速(淡入效果)。属性值设置为 ease-in 的效果等同于 cubic-bezier(0.42,0,1.0,1.0)。
- ease-out：指定一个缓慢的结束(淡出效果)。等同于 cubic-bezier(0,0,0.58,1.0)。
- ease-in-out：指定加速后再减速。等同 cubic-bezier(0.42,0,0.58,1)。
- cubic-bezier(n,n,n,n)：定义用于加速或者减速的贝塞尔曲线的形状，它们的值在

0~1 之间。

【例 14.11】

继续在上个示例的基础上进行更改，指定切换效果为加速后再减速。代码如下：

```
div{
    -webkit-transition-property: width,height;     /* 过渡名称是 width */
    -webkit-transition-duration: 2s;               /* 过渡时间为 2 秒 */
    -webkit-transition-timing-function:ease-in-out;
    /*省略其他属性设置*/
}
```

4. transition-delay属性

transition-delay 属性指定过渡效果何时开始。该属性的默认值为 0，其单位是秒或者毫秒。基本语法如下：

```
transition-delay: time;
```

【例 14.12】

继续在前面示例的基础上添加样式，在实现过渡效果之前延时 5 秒钟。代码如下：

```
div{
    -webkit-transition-property: width,height;
    -webkit-transition-duration: 2s;
    -webkit-transition-timing-function: ease-in-out;
    -webkit-transition-delay: 2s;
    /*省略其他属性*/
}
```

14.3.2 transition的简写属性

transition 是一个简写属性，用于在一个属性中设置 4 个过渡属性，这 4 个属性在上一节中已经提到。transition 属性的基本语法如下：

```
transition: property duration timing-function delay;
```

在使用 transition 属性设置过渡效果时，它的各个参数必须按照顺序进行定义，不可以进行颠倒。

例如，针对例 14.10、例 14.11 和例 14.12，可以直接通过以下代码实现：

```
div{
    /* 省略其他属性 */
    transition: width,height 2s ease-in-out 2s;
}
```

注意： 无论是单个属性还是简写属性，使用时都可以实现多个过渡效果，需要为每个过渡集中指定所有的值，并且使用逗号进行分隔。

14.4　动　　画

CSS 3 中除了支持渐变、过渡和转换特效外，还可以实现动画效果。动画是让元素从一个样式逐渐改变到另一个样式。本节首先介绍与动画有关的属性，接着介绍动画帧——@keyframes。

14.4.1　动画相关属性

通过 CSS 3，Web 开发者可以创建动画，这可以在许多网页中取代动画图片、Flash 动画以及 JavaScript 脚本。

CSS 3 中新增加了许多动画属性，这些属性及其说明如表 14-4 所示。

表 14-4　CSS 3 中新增的动画属性

属性名称	说　明
animation-name	指定@keyframes 动画的名称
animation-duration	指定动画完成一个周期所花费的秒或毫秒。默认值是 0
animation-timing-function	指定动画的速度曲线。取值说明如下。 ● linear：动画从头到尾的速度是相同的。 ● ease：动画以低速开始，然后加快，在结束前变慢。 ● ease-in：动画以低速开始。 ● ease-out：动画以低速结束。 ● ease-in-out：动画以低速开始和结束。 ● cubic-bezier(n,n,n,n)：在函数中指定自己的值，可能是从 0~1 的数值
animation-delay	指定动画何时开始。默认值是 0
animation-iteration-count	指定动画被播放的次数。默认值是 1，将属性值指定为 infinite 时，表示动画循环播放，即永远播放
animation-direction	规定动画是否在下一周期逆向地播放。默认值是 normal，表示动画每次都会正常显示；还可以指定为 alternate，表示交替逆向运动，即动画正向播放奇数次迭代，反向播放偶数次迭代
animation-play-state	规定动画是否正在运行或暂停。默认值是 running，表示动画正常运行；还可以是 paused，表示暂停动画
animation-fill-mode	规定对象动画时间之外的状态，取值说明如下。 ● none：默认值，按定义的顺序执行，且完成后返回到初始的关键帧。 ● backwards：指定关键帧在动画开始前应用样式。 ● forwards：指定关键帧在动画结束后才应用样式。 ● both：同时应用 forwards 和 backwards 的效果

除了上述属性外，还包括一个 animation，该属性是表中所有动画属性(除 animation-

play-state)的简写属性。基本语法如下：

```
animation: name duration timing-function delay iteration-count direction;
```

【例 14.13】

本例演示上述动画属性，首先需要向页面中添加一个 div 元素，并且指定该元素的 id 属性值，页面代码不再显示。接着为该 div 元素指定宽度、高度、背景、所处位置以及动画属性和多个样式属性。部分代码如下：

```
#mydiv{
    width: 100px;
    height: 100px;
    background: red;
    position: relative;
    -webkit-animation-name: myfirst;                /*动画名称*/
    -webkit-animation-duration: 5s;             /*动画完成一个周期的花费时间*/
    -webkit-animation-timing-function: linear;      /*动画的速度曲线*/
    -webkit-animation-delay: 2s;                    /*延迟时间*/
    -webkit-animation-iteration-count: infinite;    /*循环播放动画*/
    -webkit-animation-direction: alternate;         /*动画交替运动*/
    -webkit-animation-play-state: running;          /*运行动画*/
    /* 省略其他私有属性 */
}
```

上述属性可以直接通过 animation 属性和 animation-play-state 属性来代替实现。具体代码如下：

```
#mydiv{
    -webkit-animation: myfirst 5s linear 2s infinite alternate;
    -webkit-animation-play-state: running;          /*运行动画*/
    /* 省略其他属性 */
}
```

14.4.2 @keyframes

读者在浏览器中运行上述代码时可以发现，在浏览器页面除了显示一个长度和宽度为 100 像素的矩形外，没有任何别的效果。这还需要指定一个关键的属性——@keyframes。@keyframes 用于定义关键帧，一个关键帧表示动画过程中的一个状态。基本语法如下：

```
@keyframes animationname {
    keyframes-selector {css-styles;}
}
```

在上述语法中，animationname 是必需的，它定义关键帧的名称，它是一个唯一标识，其值指定 animation-name 属性的值。

keyframes-selector 也是一个必需参数，它指定当前关键帧应用到整个动画过程中的位置，该参数的值可以是 from、to 或者百分比，其中 from 和 0%的效果相同，即动画开始；to 和 100%的效果相同，即动画结束。

css-styles 也是必需的，它定义一个或多个合法的 CSS 样式属性，多个属性之间使用逗

号进行分隔。

【例 14.14】

在上个示例的基础上定义关键帧，指定动画不同阶段时的背景颜色和运行距离。代码如下：

```
@-webkit-keyframes myfirst{
    0% {background:red; left:0px; top:0px;}
    25% {background:yellow; left:600px; top:0px;}
    50% {background:blue; left:600px; top:200px;}
    75% {background:green; left:0px; top:200px;}
    100% {background:red; left:0px; top:0px;}
}
```

从上述代码中可以看出，@keyframes 与动画和过渡等相关属性一样，也需要指定私有属性。

在浏览器中运行上述代码，查看效果，初始效果如图 14-19 所示。延迟两秒钟后，动画开始运行，捕捉某一时刻的动画，效果如图 14-20 所示。

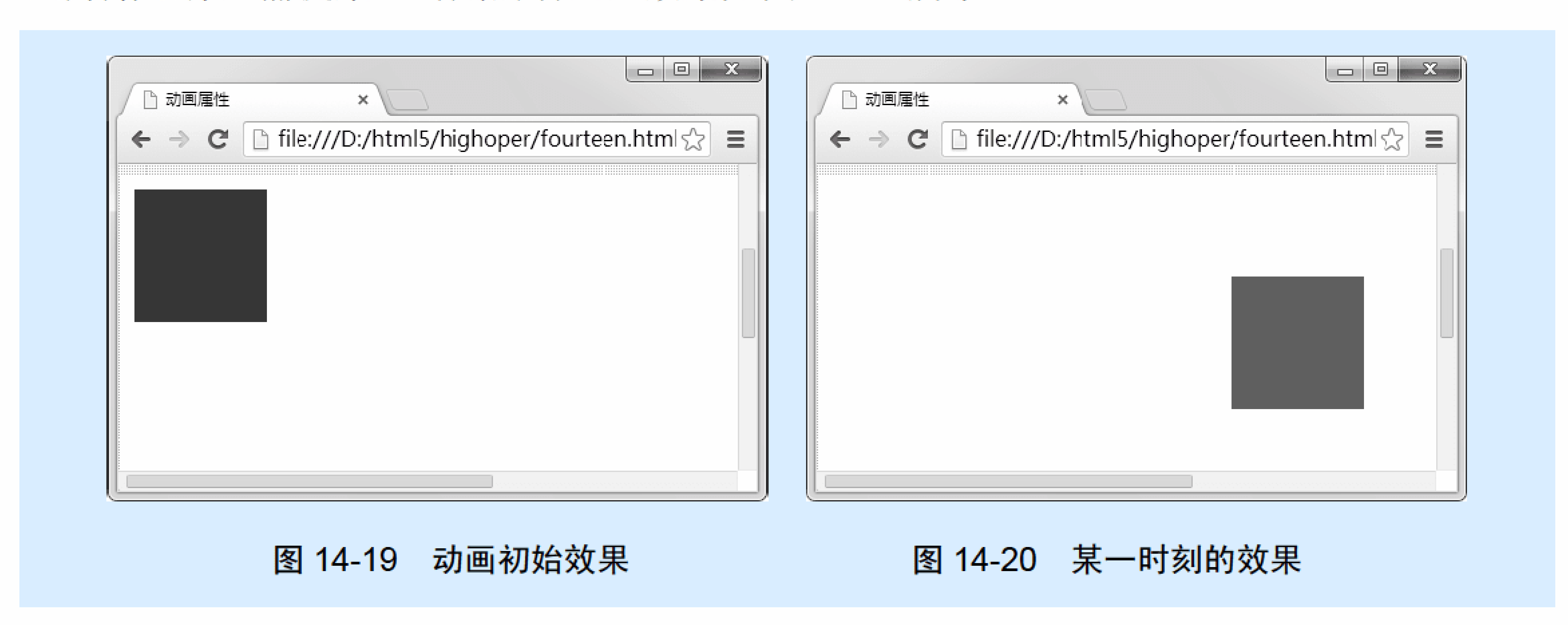


图 14-19　动画初始效果　　　图 14-20　某一时刻的效果

14.5　实战——制作动画海报圈

在本节之前，我们已经简单了解了 CSS 3 中新增的与渐变、转换和过渡有关的属性，本节实战利用先前介绍的知识实现一个比较完整的动画效果。

在制作动画海报圈时，三个环构造使用简单的 JavaScript 脚本来控制，创建元素并为它们指定转换，然后使用 CSS 动画旋转海报圈的每一环，并旋转周围的包含元素。实现步骤如下。

步骤 01　向页面中添加 h1 元素和 div 元素，使用一个 div 元素作为背景墙，里面嵌入 3 个子元素，每个子元素为海报圈的一部分。页面代码如下：

```
<h1>制作海报圈</h1>
<div id="stage">
    <div id="rotate">
        <div id="ring-1" class="ring"></div>
        <div id="ring-2" class="ring"></div>
```

```
        <div id="ring-3" class="ring"></div>
    </div>
</div>
```

步骤 02 为 id 属性值是 stage 的 div 元素定义样式，指定该元素的宽度和高度等内容。代码如下：

```
#stage {
    margin: 80px auto;
    width: 600px;
    height: 400px;
    -webkit-perspective: 800;       /*指定 3D 元素的透视效果*/
}
```

步骤 03 为 id 属性值是 rotate 的 div 元素定义样式，指定该元素的宽度、高度和动画等属性。代码如下：

```
#rotate {
    margin: 0 auto;
    width: 600px;
    height: 400px;
    -webkit-transform-style: preserve-3d;      /*确保在 3D 空间运行*/
    -webkit-animation-name: x-spin;     /*动画名称*/
    -webkit-animation-duration: 7s;     /*延迟 7 秒*/
    -webkit-animation-iteration-count: 5;   /*循环播放*/
    -webkit-animation-timing-function: linear; /*以同等速度播放动画*/
}
```

步骤 04 为 class 属性值是 ring 的 3 个 div 元素添加样式，包含宽度、高度、动画，以及第奇数个和第偶数个元素的背景颜色。内容如下：

```
.ring {
    margin: 0 auto;
    height: 110px;
    width: 600px;
    -webkit-transform-style: preserve-3d;
    -webkit-animation-iteration-count: infinite;
    -webkit-animation-timing-function: linear;
}
.ring > :nth-child(odd) {
    background-color: #FF93C9;
}
.ring > :nth-child(even) {
    background-color: #9B9BFF;
}
```

步骤 05 分别为 id 属性值是 ring-1、ring-2 和 ring-3 的 div 元素指定样式，通过 animation-name 指定动画名称，通过 animation-duration 指定动画延迟时间。代码如下：

```
#ring-1 {
    -webkit-animation-name: y-spin;
    -webkit-animation-duration: 5s;
}
```

```
#ring-2 {
    -webkit-animation-name: back-y-spin;
    -webkit-animation-duration: 4s;
}
#ring-3 {
    -webkit-animation-name: y-spin;
    -webkit-animation-duration: 3s;
}
```

步骤 06 通过@keyframes 定义名称是 x-spin 的动画，这表示整个组行使用 X 轴旋转动画时的效果。代码如下：

```
@-webkit-keyframes x-spin {
    0% {-webkit-transform: rotateX(0deg);}
    50% {-webkit-transform: rotateX(180deg);}
    100% {-webkit-transform: rotateX(360deg);}
}
```

步骤 07 分别通过@keyframes 定义名称是 y-spin 和 back-y-span 的动画，它们表示整个组行使用 Y 轴旋转动画时的效果。代码如下：

```
@-webkit-keyframes y-spin {
    0% {-webkit-transform: rotateY(0deg);}
    50% {-webkit-transform: rotateY(180deg);}
    100% {-webkit-transform: rotateY(360deg);}
}
@-webkit-keyframes back-y-spin {
    0% {-webkit-transform: rotateY(360deg);}
    50% {-webkit-transform: rotateY(180deg);}
    100% {-webkit-transform: rotateY(0deg);}
}
```

步骤 08 添加 JavaScript 脚本代码，首先在外部声明两个常量，然后通过 setup_posters()函数定义海报。代码如下：

```
const POSTERS PER ROW = 12;     //定义海报的行
const RING RADIUS = 200;        //定义圆角特效
function setup posters (row) {
    var posterAngle = 360 / POSTERS PER ROW;
    for (var i=0; i<POSTERS PER ROW; i++) {
        var poster = document.createElement('div');
        poster.className = 'poster';
        // 计算和分配海报的转换
        var transform = 'rotateY(' + (posterAngle * i)
          + 'deg) translateZ(' + RING RADIUS + 'px)';
        poster.style.webkitTransform = transform;
        //设置显示在海报中的数字
        var content = poster.appendChild(document.createElement('p'));
        content.textContent = i;
        row.appendChild(poster);        //海报添加到行
    }
}
```

步骤 09 继续添加 JavaScript 脚本，页面加载完成后分别调用 setup_posters()函数，

并传入不同的 div 子元素。代码如下：

```
window.onload = function(){
    setup posters(document.getElementById('ring-1'));
    setup posters(document.getElementById('ring-2'));
    setup posters(document.getElementById('ring-3'));
}
```

步骤 10 添加元素的其他样式或私有属性特效，例如步骤 08 中 class 属性值是 poster 的样式，这里省略其他代码。

步骤 11 所有的页面代码和样式代码添加完毕后，运行页面，页面运行时已经开始动画。为了演示静态效果，可以首先注释掉@keyframes 属性的设置，此时的效果如图 14-21 所示。

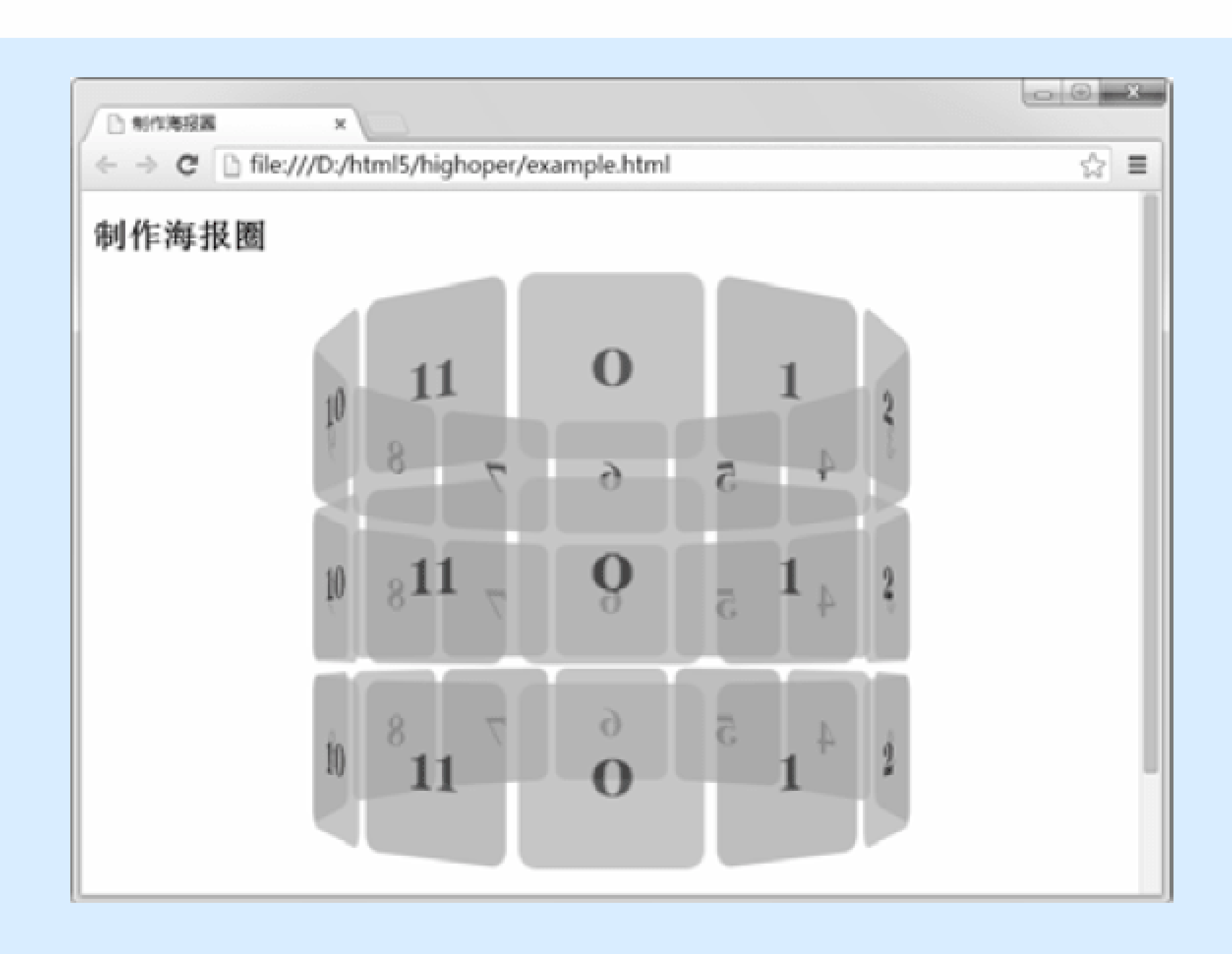


图 14-21　海报圈的静态效果

步骤 12 取消对@keyframes 属性的注释，然后刷新页面观察动画，图 14-22 显示了某一时刻海报圈的动画效果。



图 14-22　海报圈某一时刻的动画效果

14.6 本章习题

1. 填空题

(1) 实现渐变效果时，使用 Webkit 引擎的浏览器需要添加__________私有属性。
(2) __________是指从起点到终点、颜色从内到外进行的圆形渐变。
(3) 在实现过渡效果时，transition-duration 属性的默认值是__________。
(4) __________属性将与过渡有关的 4 个属性进行了简写。
(5) animation-play-state 属性的值设置为__________时，表示动画正常运行。

2. 选择题

(1) 在下面的方法中，__________方法定义 2D 转换，沿着 X 轴和 Y 轴移动元素。

A. transform(x,y)　　B. translate(x,y)
C. translateX(n)　　D. translateY(n)

(2) __________属性用于设置过渡效果的名称。

A. transition-property　　B. transition-duration
C. transition-timing-function　　D. transition-delay

(3) 当 transition-timing-function 属性取值为__________时，表示指定一个缓慢的开始，然后逐渐加速(淡入效果)。

A. ease　　B. ease-in
C. ease-out　　D. ease-in-out

(4) __________用于定义动画的名称。

A. animation-timing-function　　B. animation-duration
C. animation-name　　D. animation-property

(5) animation-fill-mode 属性指定对象动画时间之外的状态，默认值为__________。

A. forwards　　B. none
C. backwards　　D. both

3. 上机练习

(1) 为相框指定渐变和阴影效果

利用本章介绍的内容实现一个渐变效果，为一张图片添加边框，同时指定边框的阴影效果和渐变颜色，实现效果如图 14-23 所示。

如下代码为渐变颜色:

```
background: -webkit-linear-gradient(right, rgba(255,255,255,0),
  rgba(255,255,255,1)), url(insert.jpg);
```

(2) 实现 3D 图片墙翻转效果

根据本章的实战模拟实现一个 3D 图片墙的翻转效果，每一行图片在不同的时间周期运行不一致的翻转动画，某一时刻的效果如图 14-24 所示。



图 14-23　上机练习(1)的效果

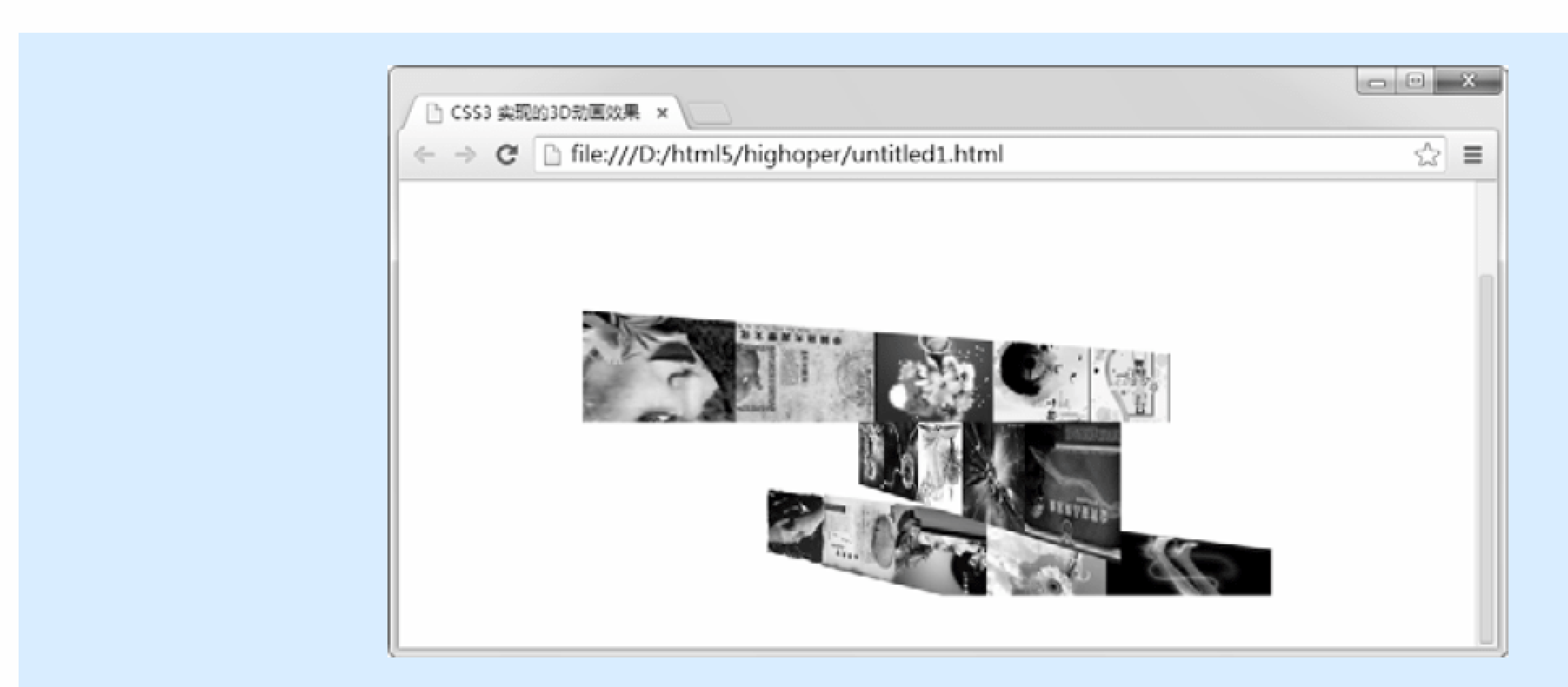


图 14-24　3D图片墙翻转效果





第 15 章

HTML 5 + CSS 3 页面案例

本书前面的 14 章详细介绍了 HTML 5 技术、CSS 3 技术以及 JavaScript 技术，本章结合这 3 种技术，实现页面的综合效果。

本章学习目标：

- 了解贪吃蛇小游戏的设计
- 掌握页面贪吃蛇的实现
- 掌握 HTML 5 技术与 JavaScript 技术的结合
- 掌握 HTML 5 技术与 jQuery 技术的结合
- 掌握 HTML 5 技术与 CSS 3 技术的结合
- 掌握 CSS 3 技术与 JavaScript 技术的结合
- 掌握常见的几种页面特效

15.1　JavaScript 经典贪吃蛇

贪吃蛇是一款经典的小游戏，通常在页面中使用 Flash 来实现。在 HTML 5 技术出现以后，该游戏可以通过 HTML 5 技术与 JavaScript 技术相结合的方式，在页面中实现。

本节详细介绍贪吃蛇的页面实现，使用 HTML 5 技术与 JavaScript 技术完成贪吃蛇的游戏设计，并通过 CSS 3 技术对页面进行美化。

15.1.1　案例分析

贪吃蛇是一款可以在网页中实现的小游戏，其游戏规则是通过按键控制蛇爬行的转弯方向，使蛇头接触游戏中随机产生的食物。

(1)　页面中需要实现的事项如下：

- 在页面的任意位置随机地出现食物。
- 在页面的固定位置出现有一定长度的蛇并直线移动。
- 使用键盘的上、下、左、右按键控制蛇转身。
- 当蛇的头部接触食物时，被视为蛇吃掉了食物，蛇的身体长度增加，并再次随机出现新的食物。
- 当蛇撞到墙(游戏区域的边缘)的时候，其头部在墙的另一端出现(或提示失败，结束游戏)。
- 当蛇的头部撞到其自身的时候，提示失败，游戏结束。

(2)　这个游戏程序中，HTML 5 和 JavaScript 结合很紧密，需要掌握 HTML 5 和 JavaScript 的基础知识。上述游戏规则中，可以把这些事项定义为 6 个函数。

- 蛇的移动：包括定义蛇的出现位置、长度、颜色，蛇的速度、爬行路线等。
- 蛇撞墙：本案例当蛇撞墙时，蛇从游戏区域的另一端回到游戏区域。因此蛇头位置在边界时，需要改变蛇头位置。
- 改变蛇的方向：根据按键改变蛇头的位置。
- 放置食物：在游戏区域内的随机位置产生食物。
- 吃到食物：增加蛇的长度、放置新的食物。
- 游戏结束：当蛇的头部撞到其自身的时候，提示失败，游戏结束。

HTML 5 中有了 canvas 元素和绘制矩形的函数，可以根据 canvas 元素设计游戏的区域，并在该区域内绘制矩形，分别绘制蛇和食物。通过 JavaScript 控制矩形的位置，实现蛇的移动和食物的出现。

15.1.2　JavaScript实现

该游戏使用 HTML 5 和 JavaScript 技术，其核心是在页面中添加 canvas 元素，通过 JavaScript 在 canvas 元素的区域内绘制矩形并控制矩形。因此可以向页面中添加 canvas 元素并定义其 id 为“myCanvas”；定义其长和宽都为 400。

接下来是 JavaScript 的实现。在本章 15.1.1 小节中将游戏分为 6 个函数，但页面开始时需要执行的只有放置食物和蛇的移动；在游戏运行期间只能够改变蛇的方向。因此可将蛇撞墙、吃到食物和吃到自己这几个函数放在蛇的移动函数中。这些函数将在蛇移动的同时被依据条件选择性地执行。

在 JavaScript 中实现贪吃蛇的步骤如下。

步骤 01 首先在<script>标记下定义蛇的基本数据，分别定义蛇的速度、初始位置、食物位置、蛇的长度、爬行路径、数据单位和位移改变量，代码如下：

```
var c = document.getElementById("myCanvas");
var time = 300; //蛇的速度
var cxt = c.getContext("2d");
var x = y = 32;
var a = 0; //食物坐标
var t = 8; //蛇身长
var map = []; //记录蛇运行路径
var size = 8; //蛇身单元大小
var direction = 2; //位置改变量：1 向上  2 向右  0 向左  3 向下
```

步骤 02 接下来定义食物的放置，其实质是在 canvas 元素区域内的随机位置绘制矩形。由于食物在游戏开始的时候就需要显示，因此在定义了该函数之后需要直接在<script>标记下运行，代码如下：

```
function rand frog() {
    a = Math.ceil(Math.random() * 50);
    cxt.fillStyle = "#00B100"; //内部填充颜色
    cxt.strokeStyle = "#00B100"; //边框颜色
    cxt.fillRect(a * 8, a * 8, 8, 8); //绘制矩形
}
rand_frog();
```

步骤 03 最后是比较重要的环节：蛇的移动和改变方向。蛇的移动需要使用蛇的位置改变量，而这个量在改变蛇的方向时会发生变化。首先定义蛇的改变方向函数，改变蛇的位置改变量，代码如下：

```
document.onkeydown = function (e) { //改变蛇方向
    var code = e.keyCode - 37;
    switch (code) {
        case 1: direction = 1; break; //上
        case 2: direction = 2; break; //右
        case 3: direction = 3; break; //下
        case 0: direction = 0; break; //左
    }
}
```

步骤 04 蛇的移动只需要根据蛇的位置改变量来改变蛇头坐标，并根据蛇的长度绘制矩形；在蛇移动过程中，保持蛇的长度，清除移动时蛇尾的矩形块。

这个过程中有蛇撞墙的事件、吃食物的事件和咬到自己的事件。排除这几个事件，蛇的移动代码如下：

```
function set_game_speed() { // 移动蛇
```

```
    switch (direction) {
        case 1: y = y - size; break;
        case 2: x = x + size; break;
        case 0: x = x - size; break;
        case 3: y = y + size; break;
    }
    if (map.length > t) { //保持蛇身长度，去掉尾部矩形块
        var cl = map.shift(); //删除数组第一项，并且返回原元素
        cxt.clearRect(cl['x'], cl['y'], size, size);
    }
    map.push({ 'x': x, 'y': y }); //将数据添加到原数组尾部
    cxt.fillStyle = "#00B100"; //内部填充颜色
    cxt.strokeStyle = "#00B100"; //边框颜色
    cxt.fillRect(x, y, size, size); //绘制矩形
}
```

步骤 05 解决碰壁问题。蛇的碰壁发生在其蛇头坐标处在 canvas 元素区域的边界位置时，由于 canvas 元素的长和宽都是 400，因此当蛇头坐标变量 x 和 y 为 400 或 0 时将碰壁，此时只需将碰壁的水平坐标或垂直坐标改为相反位置的临界即可，代码如下：

```
if (x > 400 || y > 400 || x < 0 || y < 0) { //解决碰壁问题
    if (x > 400)
    { x = 0; }
    if (y > 400)
    { y = 0; }
    if (x < 0)
    { x = 400; }
    if (y < 0)
    { y = 400; }
}
```

上述代码需要放置在步骤 04 的函数内。

步骤 06 蛇咬到自己的情况，需要使用循环语句来判断。根据比较蛇的路径和蛇头坐标来确定。当蛇头坐标与蛇的路径坐标有重复时，则表示蛇咬到了自己，提示失败并刷新页面，代码如下：

```
for (var i=0; i<map.length; i++) {
    if (parseInt(map[i].x)==x && parseInt(map[i].y)==y) {
        alert("咬到自己了.....");
        window.location.reload();
    }
}
```

上述代码需要放置到步骤 04 的函数内。

步骤 07 蛇吃到食物的情况，是指蛇头位置与食物位置重复，需要水平坐标与垂直坐标同时相等，此时需要增加蛇的长度，并绘制新的食物，代码如下：

```
if ((a*8)==x && (a*8)==y) { //吃食物
    t++;
    rand_frog();
}
```

上述代码需要放置在步骤 04 的函数内。

步骤 08 上述代码已经实现了蛇的移动，由于蛇的移动是在游戏开始时就执行，并在指定的时间间隔下重复执行，因此需要在<script>标记下添加语句，在指定时间间隔下执行蛇的移动函数 set_game_speed()，代码如下：

```
interval = window.setInterval(set_game_speed, time); // 移动蛇
```

步骤 09 为了显示 canvas 的区域，为 canvas 添加背景色，其效果如图 15-1 所示，按键控制了蛇的转向。在蛇吃了几次食物后，其长度明显增加，如图 15-2 所示。

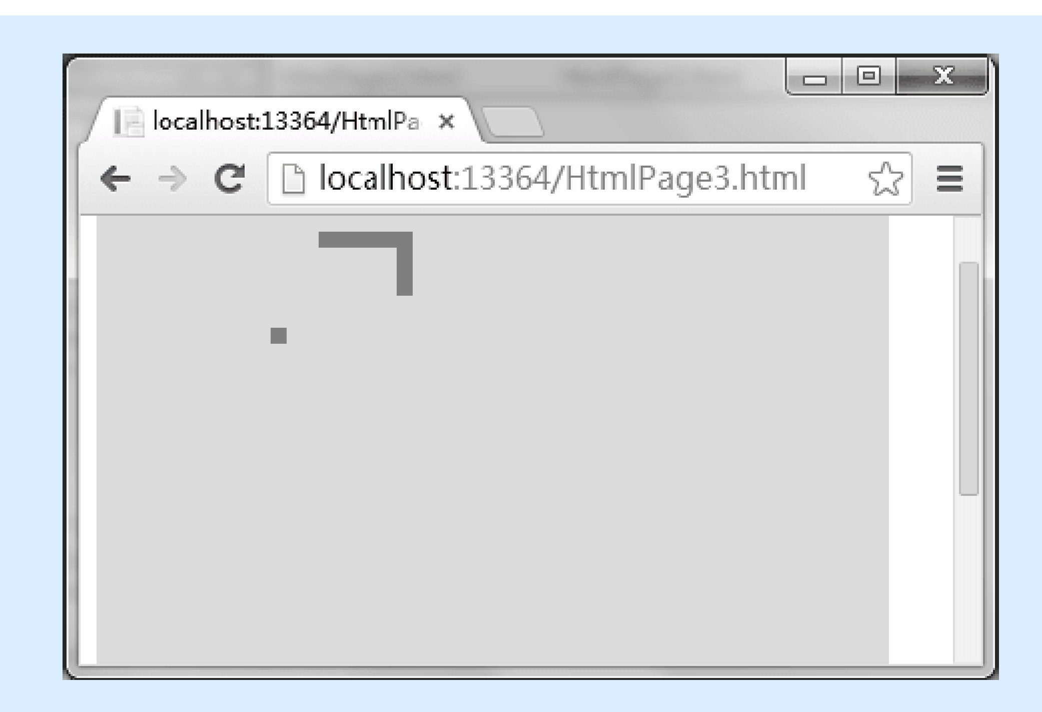


图 15-1　蛇的初始大小

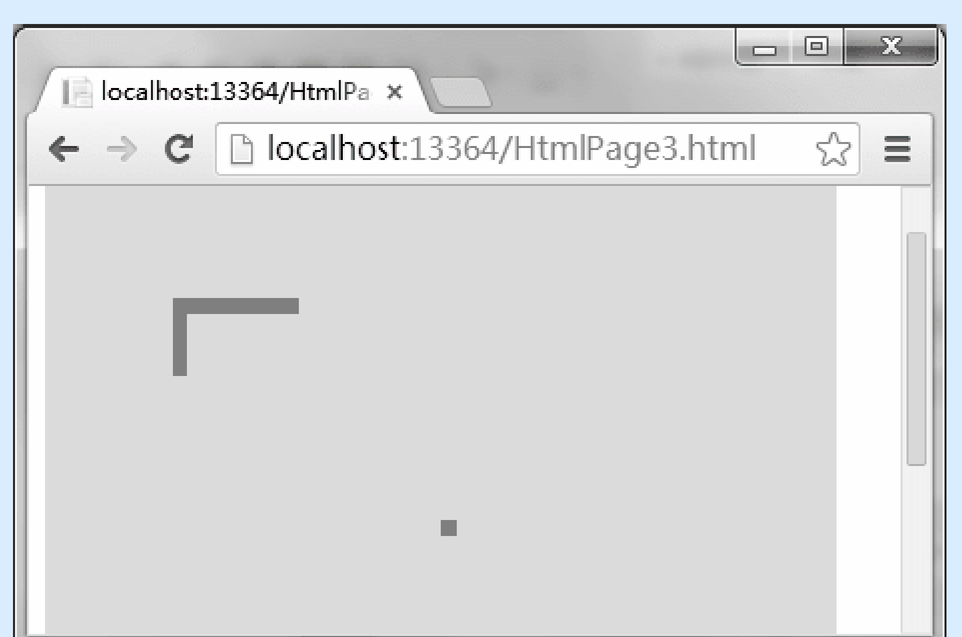


图 15-2　蛇吃了几次食物后

15.1.3 页面美化

上面通过 HTML 5 与 JavaScript 的结合实现了贪吃蛇小游戏，但该游戏页面简单，没有布局和样式，页面显得单调。这里我们将使用 div 样式，为该游戏添加边框和布局，步骤如下。

步骤 01 首先为页面添加 div 的定位，要求向左浮动，代码如下：

```
div {
    float: left;
}
```

步骤 02 为页面添加头部的 div，为其添加背景图片和标题“小游戏”。头部的 div 需要占满一行并居中显示，其背景图片不能重复并铺满 div，格式代码如下：

```
.div1 {
    width: 100%;
    background-image: url('Images/wood.jpg');
    background-repeat: no-repeat;
    background-size: 100%;
}
```

步骤 03 头部以下的部分设置为左、中、右 3 个 div，都使用靠左浮动。左右两端的 div 不添加内容，只起到占位的作用，可为其设置百分比宽度。中间的 div 中放置 15.1.2 小节中的 canvas 元素(即小游戏)。该区域使用白色背景，使用清新的边框背景，并同时为 canvas 添加边框。

由于小游戏的区域大小是固定的，因此可以使用固定大小的 div，根据边框的宽度为 canvas 设置上边距。其代码如下：

```
.div2 {
    border-image: url('Images/flower.jpg');
    border-image-width: 55px;
    border-image-slice: 10%;
    text-align: center;
    vertical-align: middle;
    background-color: #FFFFFF;
    width: 609px;
    height: 510px;
}

canvas {
    position: relative;
    top: 55px;
    background-color: #F0DB98;
    border: 1px solid #c3c3c3;
}
```

步骤 04　上述代码省略了左右两个 div 的内容和样式，用户可自行添加。如添加导航链接、其他游戏的列表或图表等。执行效果如图 15-3 所示。

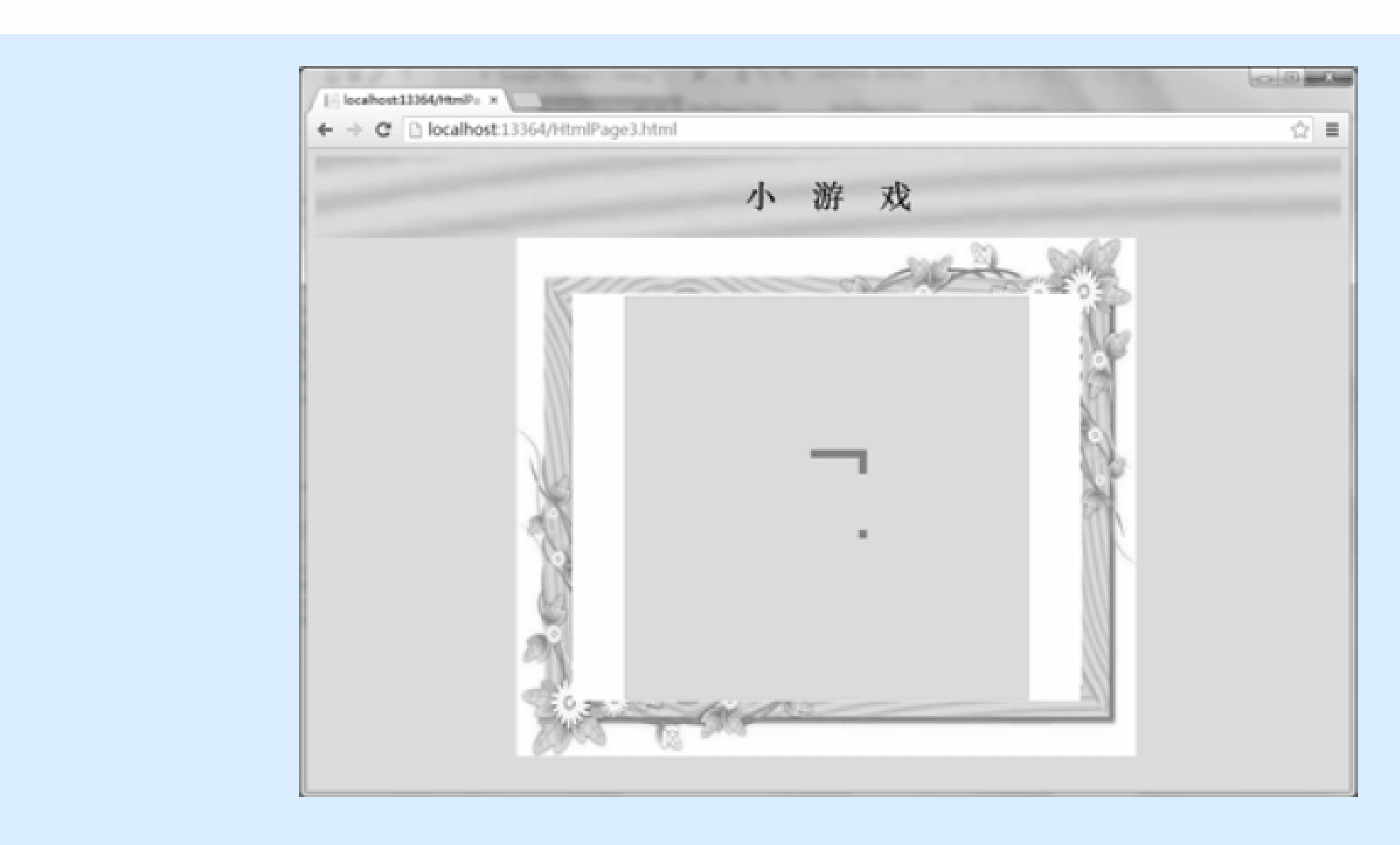


图 15-3　贪吃蛇页面美化

15.2　jQuery 导航特效

本章 15.1 节主要介绍了 HTML 5 和 JavaScript 结合实现的小游戏，本节主要介绍由 CSS 3 和 jQuery 结合实现的导航特效。

15.2.1　jQuery简介

jQuery 是继 prototype 之后又一个优秀的 JavaScript 框架。它是轻量级的 js 库，它兼容

CSS 3，还兼容各种浏览器(IE 6.0、FF 1.5、Safari 2.0、Opera 9.0)。

jQuery 2.0 及后续版本将不再支持 IE 6/7/8 浏览器。jQuery 使用户能更方便地处理 HTML、events、实现动画效果，并且方便地为网站提供 Ajax 交互。jQuery 还有一个比较大的优势，即它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的插件可供选择。jQuery 能够使用户的 HTML 页面保持代码和内容分离，也就是说，不用再在 HTML 里面插入一堆 js 来调用命令了，只需定义 id 即可。

jQuery 是一个兼容多浏览器的 JavaScript 框架，核心理念是“write less，do more”(写得更少，做得更多)。jQuery 在 2006 年 1 月由美国人 JohnResig 在纽约的 barcamp 发布，吸引了来自世界各地的众多 JavaScript 高手加入，由 DaveMethvin 率领团队进行开发。如今，jQuery 已经成为最流行的 JavaScript 框架，在世界前 10000 个访问最多的网站中，有超过 55%在使用 jQuery。

jQuery 是免费、开源的，使用 MIT 许可协议。jQuery 的语法设计可以使开发更加便捷，例如操作文档对象、选择 DOM 元素、制作动画效果、事件处理、使用 Ajax 以及其他功能。除此以外，jQuery 提供 API，让开发者可以编写插件。其模块化的使用方式使开发者可以很轻松地开发出功能强大的静态或动态网页。

在使用 jQuery 的 js 库之前，需要向页面中添加如下语句：

```
<script src="http://www.codefans.net/ajaxjs/jQuery-1.6.2.min.js">
</script>
```

15.2.2 jQuery实现导航特效

本节使用 jQuery 实现伸缩导航条，通过 CSS 为导航链接设置链接单击前后的样式，并通过 jQuery 将这些样式动态地显示出来。

本节所介绍的导航以链接列表的形式来呈现，导航菜单为游戏的分类和各个类别下的游戏名称，如图 15-4 所示。

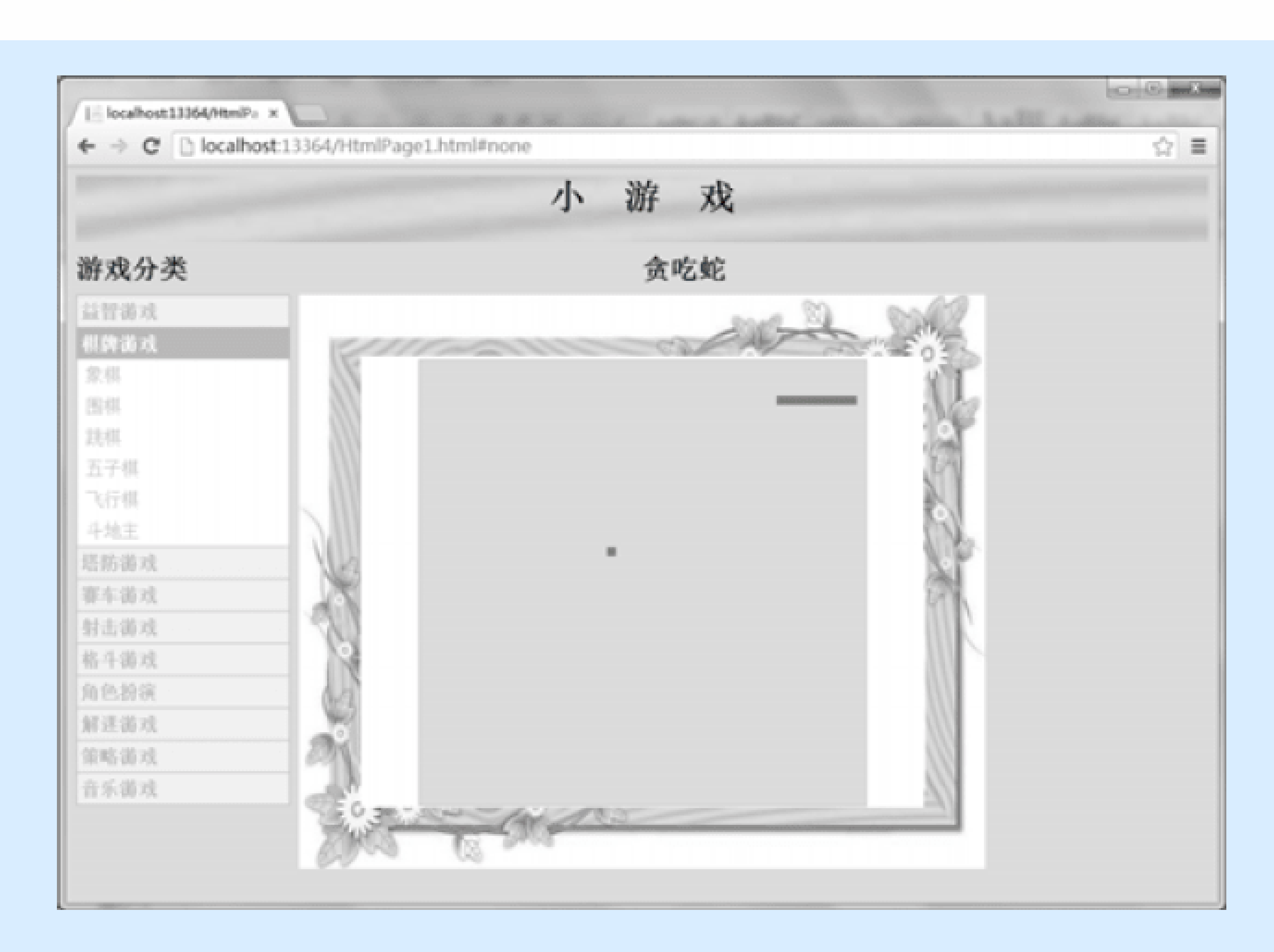


图 15-4　导航特效

图 15-4 中，导航菜单下有游戏分类，而各个分类下有子菜单。这些子菜单以链接的形式呈现。

使用 HTML 中的 a 元素和列表元素。先定义链接样式，接下来定义 jQuery 语句，步骤如下。

步骤 01 首先定义链接的样式，链接的一般样式代码如下：

```
a {
    color: #f16f0f;
    text-decoration: none;
}
```

步骤 02 定义鼠标悬停在链接上的样式，代码如下：

```
a:hover {
    color: #f16f0f;
    text-decoration: underline;
}
```

步骤 03 导航菜单以列表嵌套的形式来显示，定义外层列表的样式，代码如下：

```
.menu {
    margin-right: 8px;
    overflow: hidden;
    border-color: #DBBB66;
    border-style: solid;
    border-width: 0 1px 1px;
}
.menu li ul {
    overflow: hidden;
}
```

步骤 04 定义导航在鼠标单击前(展开前)的样式，即外层列表项的样式，具体代码如下：

```
.menu li.level1 a {
    display: block;
    height: 28px;
    line-height: 28px;
    background: #F4F4D0;
    font-weight: 700;
    color: #DBBB66;
    text-indent: 4px;
    border-top: 1px solid #DBBB66;
}
.menu li.level1 a:hover {
    text-decoration: none;
}
.menu li.level1 a.current {
    color: #FFFFFF;
    background: #EAB45A;
}
```

步骤 05 定义内层列表的样式，代码如下：

```
.menu li ul.level2 {
    display: none;
}
.menu li ul.level2 li a {
    display: block;
    height: 28px;
    line-height: 28px;
    background: #ffffff;
    font-weight: 400;
    color: #DBBB66;
    text-indent: 8px;
    border-top: 0px solid #DBBB66;
    overflow: hidden;
}
.menu li ul.level2 li a:hover {(这个什么都没有吗？)
}
```

步骤 06 上述 5 个步骤定义了导航的所有样式，接下来在页面中添加 jQuery 的 js 库引用，并添加 jQuery 代码，实现当前元素的子元素呈现，同时合并其他元素的子元素，代码如下：

```
<script src="http://www.codefans.net/ajaxjs/jQuery-1.6.2.min.js">
</script>
<script>
    $(document).ready(function () {
        $(".level1 > a").click(function () {
            $(this).addClass("current")  //给当前元素添加 current 样式
            .next().show()  //下一个元素显示
            //父元素的兄弟元素的子元素<a>移除 current 样式
            .parent().siblings().children("a").removeClass("current")
            .next().hide();  //它们的下一个元素隐藏
            return false;
        });
    });
</script>
```

步骤 07 向页面中添加列表，使用先前定义的 CSS 样式。由于篇幅有限，本步骤只提供一条列表的添加代码，代码如下：

```
<ul class="menu">
    <li class="level1">
        <a href="#none">益智游戏</a>
        <ul class="level2">
            <li><a href="#none">拼图</a></li>
            <li><a href="#none">迷宫</a></li>
            <li><a href="#none">祖玛</a></li>
            <li><a href="#none">连连看</a></li>
            <li><a href="#none">密室逃脱</a></li>
            <li><a href="#none">消消看</a></li>
        </ul>
    </li>
</ul>
```

步骤 08 将上述代码放在本章 15.1.3 小节中页面的左侧 div 中，其效果如图 15-5 所

示。单击“棋牌游戏”导航菜单，其子菜单如图 15-4 所示。

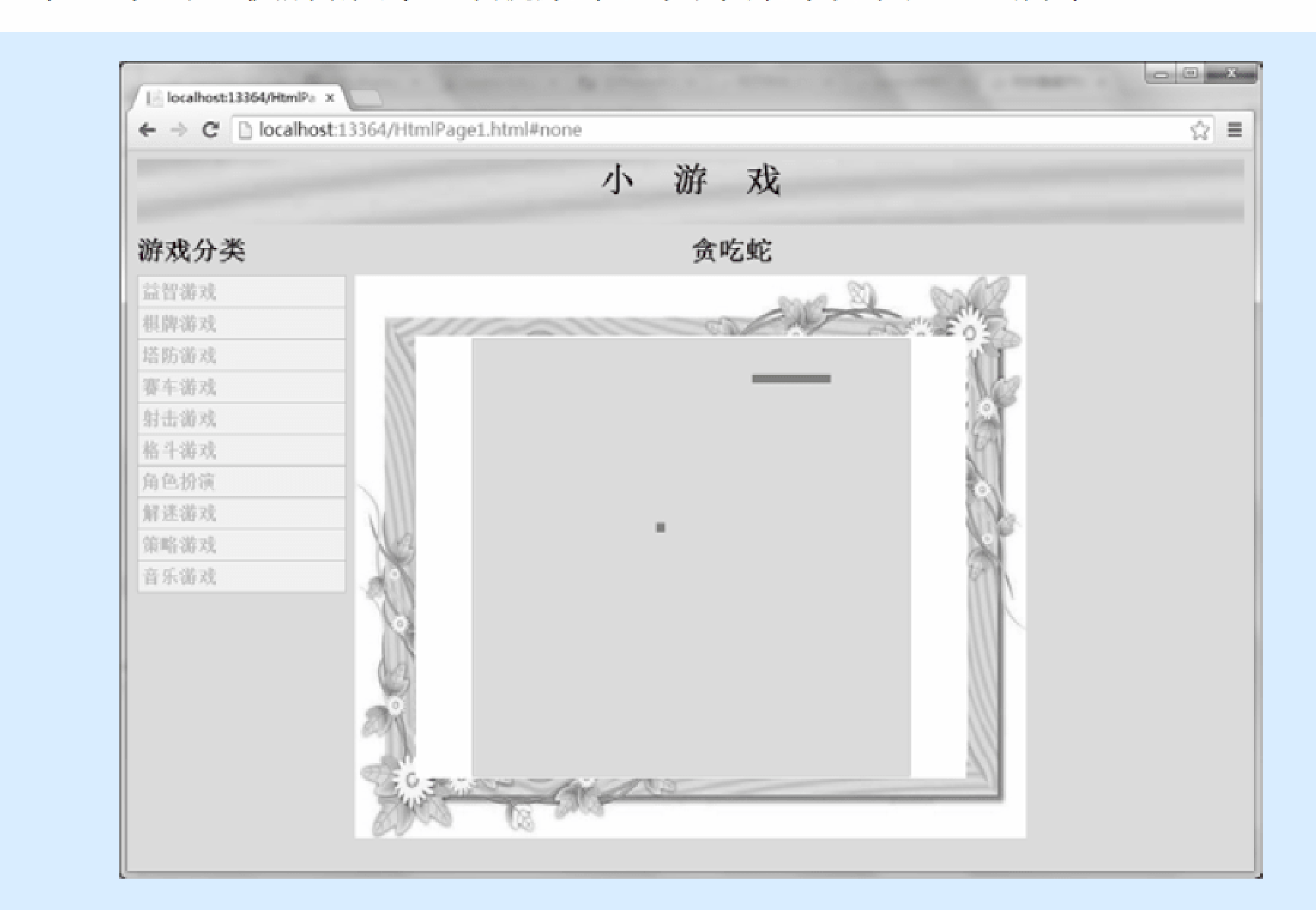


图 15-5　导航样式

15.3　CSS 3 图片特效

本章 15.1 节和 15.2 节分别介绍了 HTML 5 技术与 JavaScript 技术的结合、CSS 3 技术和 jQuery 技术的结合，本节将通过 HTML 5 技术与 CSS 3 技术的结合，实现图片的特效。

本节使用 CSS 3 技术对页面中的图片进行放大(切换)操作。网页中由于页面尺寸有限，一些需要显示图片的地方使用小图片来呈现，但小图片通常不能够满足用户需求。因此可以对图片进行样式设置，使图片在用户鼠标悬浮的时候执行图片的放大或切换。

本节介绍的图片特效是实现小图片的切换。当用户鼠标放在小图片上时，该图片上方将重叠一个大的图片，对小图片进行解释说明。大图片可以是小图片的放大图片；也可以是对小图片的介绍。

如游戏图片，在鼠标悬浮时可呈现该游戏的画面，本节在图 15-5 的基础上，在页面的右侧 div 中添加小游戏的图标，并在用户鼠标悬浮图标时呈现该游戏的画面。图标以链接的形式呈现，可单击进入该游戏的页面。使用 CSS 3 和 HTML 5 的步骤如下。

步骤 01　首先定义该 div 的样式，定义文本没有缩进，并且有着 14px 的边距，代码如下：

```
.divp {
    text-indent: 0;
    margin-left: 14px;
}
```

步骤 02　定义图标链接在鼠标悬浮前的样式，代码如下：

```
.thumbnail {
   position: relative;
   z-index: 0;
}
```

步骤 03 定义图标链接在鼠标悬浮时的样式，代码如下：

```
.thumbnail:hover {
   background-color: transparent;
   z-index: 50;
}
```

步骤 04 定义图标在被鼠标悬浮时，其显示的重叠图片的样式。为图片添加边框并添加说明文字，使用 span 元素。其样式代码如下所示：

```
.thumbnail span {
   position: absolute;
   background-color: lightyellow;
   left: -430px;
   border: 1px dashed gray;
   visibility: hidden;
   color: black;
   text-decoration: none;
}
.thumbnail span img {
   border-width: 0;
   padding: 2px;
}
```

步骤 05 定义 span 元素在鼠标悬浮时显示，代码如下：

```
.thumbnail:hover span {
   visibility: visible;
   top: 0;
   right: 1px;
}
```

步骤 06 为右侧的 div 添加样式，可列表显示游戏图标。图标的大图放在 span 元素中显示。由于篇幅有限，这里提供一条列表的代码，一条列表显示两个游戏图标，代码如下：

```
<div class="divp">
   <ul>
      <li><a class="thumbnail" href="#thumb">
            <img src="Images/game1s.jpg" border="0" />
            <span>
               <img src="Images/game11.jpg" border="0" />
                  经典塔防游戏~</span></a>
          
         <a class="thumbnail" href="#thumb">
            <img src="Images/game2s.jpg" border="0" />
            <span>
               <img src="Images/game21.jpg" border="0" />
                  超萌塔防游戏~</span>
```

```
            </a>
        </li>
    </ul>
</div>
```

步骤 07 上述代码的执行效果如图 15-6 所示。



图 15-6 小游戏的图标列表

步骤 08 将鼠标放在小游戏的图标上面(如“保卫萝卜”的图标)，将在页面显示该图标的游戏画面，如图 15-7 所示。



图 15-7 图片放大特效

15.4　其他页面效果

网页中的样式效果有很多，通过 HTML 5 技术、CSS 3 技术以及 JavaScript 技术来实现。本节介绍几种常用的页面效果。

15.4.1　页面悬浮广告

大多网站页面的左右两侧，都有悬浮的矩形小广告，广告可以展开、可以隐藏。在如图 15-7 所示页面中添加左侧的广告框，显示“关闭”和“展开”按钮来隐藏和展开图片。

页面的悬浮广告是通过简单的 JavaScript 来实现的，其核心是添加可以重叠的 div 块，并通过两个按钮切换 div 块。

广告在展开和关闭状态时显示不同的 div，两个 div 使用不同的广告图片(关闭状态下显示的是原图片右侧截取部分)，步骤如下。

步骤 01　定义展开状态下的 div 和关闭链接，代码如下所示：

```
<div id="Bar1190 big" style="position: absolute; z-index: 9; top: 10px;
  left: 0px; width: 160px; height: 284px; margin-top:200px">
    <div id="AD1190" style="width: 160px; height: 284px; text-align:
      center; float: none" class = "adSpace">
        <a href="/" target=" blank">
            <img width="160" height="284"
              src="Images/guanggao.jpg" alt="" border="0"></a>
    </div>
    <div style="height: 18px; width: 100px; background: #CCCCCC;
      text-align: right; line-height: 18px;">
        <a style="font-size: 12px; cursor: pointer;"
          onclick="bar1190 hidden()">关 闭</a>
    </div>
</div>
```

步骤 02　定义关闭状态下的 div 和展开链接，代码如下：

```
<div id="Bar1190 small" style="position: absolute; z-index: 9; top: 10px;
display: none; left: 0px; width: 22px; height: 284px; margin-top:200px">
     <div id="Div1" style="width: 22px; height: 284px; text-align: center;
       float: none" class="adSpace">
         <a href="/" target=" blank">
             <img width="49" height="283" src="Images/guanggaos.jpg"
               alt="" border="0"></a>
     </div>
     <div style="height: 18px; width: 25px; background: #CCCCCC;
       text-align: right; line-height: 18px;">
         <a style="font-size: 12px; cursor: pointer;"
           onclick="bar1190 show()">展开</a>
     </div>
</div>
```

步骤 03　定义 JavaScript 脚本，切换展开状态和关闭状态，代码如下：

```
<script type="text/JavaScript">
    function bar1190 show() {
        document.getElementById('Bar1190 big').style.display = '';
        document.getElementById('Bar1190 small').style.display = 'none';
    }
    function bar1190 hidden() {
        document.getElementById('Bar1190 big').style.display = 'none';
        document.getElementById('Bar1190 small').style.display = '';
    }
    var autohide1190 = setTimeout("bar1190 hidden()", 6000);
</script>
```

步骤 04 将上述代码放在如图 15-7 所示的页面中，其效果如图 15-8 所示。



图 15-8　悬浮广告的展开状态

步骤 05 如图 15-8 所示，广告窗悬浮在页面左侧。单击悬浮广告的关闭链接，其效果如图 15-9 所示。



图 15-9　悬浮广告的关闭状态

15.4.2 鼠标特效

鼠标特效也是页面中的常用特效，如在鼠标移动时，有着跟随鼠标浮动的文字或图形、在鼠标悬浮在指定位置时出现不同的样式等。

鼠标的特效需要使用鼠标事件，在本书第 3 章中有介绍。除了鼠标单击事件以外，鼠标特效还包括鼠标移动事件、鼠标移到某元素之上的事件和鼠标移开事件。

- onmousemove：鼠标被移动。
- onmouseout：鼠标从某元素移开。
- onmouseover：鼠标移到某元素之上。

根据上述鼠标事件，为如图 15-9 所示的页面添加鼠标特效，设置当鼠标放在头部“小游戏”所在的 div 时，取消该 div 的背景图片；当鼠标离开该 div 时，恢复该 div 的背景图片，步骤如下。

步骤 01 定义 div 的鼠标移入时的函数，修改其背景图片地址为空字符串：

```
function over() {
    document.getElementById("div1").style.backgroundImage = 'url("")';
}
```

步骤 02 定义 div 的鼠标移出时执行的函数，修改其背景图片地址为原地址：

```
function out() {
    document.getElementById("div1").style.backgroundImage =
      'url("Images/wood.jpg")';
}
```

步骤 03 修改该 div 的 id 属性并添加其鼠标移入事件和鼠标移出事件，代码如下：

```
<div id="div1" class="div1" onmouseover="over()" onmouseout="out()">
```

步骤 04 运行该页面，当鼠标移到头部的 div 时，其执行效果如图 15-10 所示。当鼠标移出时，该 div 背景图片恢复，如图 15-9 所示。

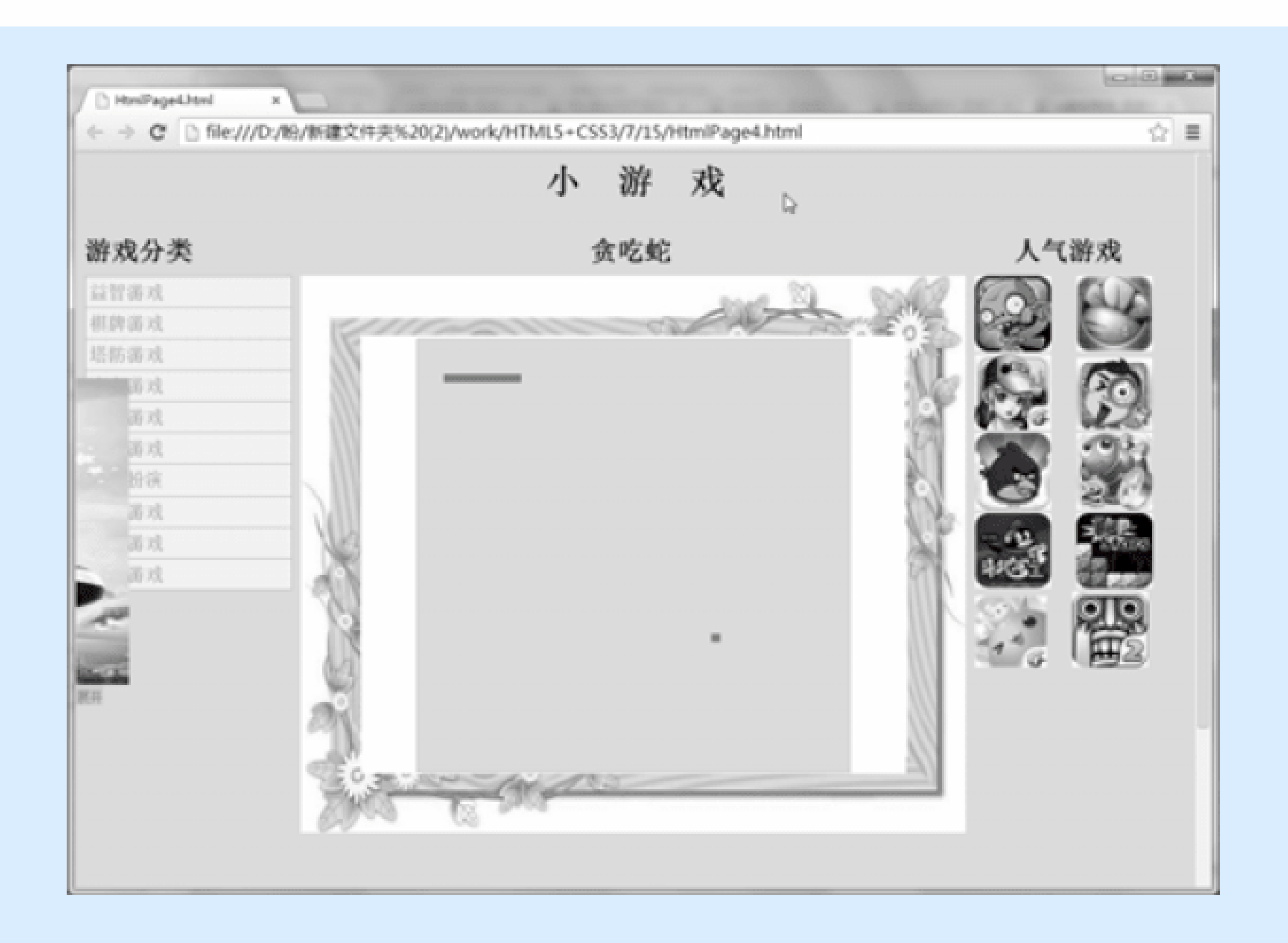


图 15-10 鼠标移入事件